TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

Diplomityö
1.12.2004

Tero Tähtinen

HTTP-VÄLITYSPALVELIMEN KÄYTTÖ TAPAHTUMIEN KERÄÄMISEEN

Työn valvoja Petri Vuorimaa

Työn ohjaaja Timo Engblom

TEKNILLINEN KORKEAKOULU Tietotekniikan osasto	DIPLOMITYÖN TIIVISTELMÄ

Tekijä	Päiväys
Tero Tähtinen	1.12.2004
	Sivumäärä 89

Työn nimi

HTTP-välityspalvelimen käyttö tapahtumien keräämiseen

Professuuri	Koodi
Vuorovaikutteinen digitaalinen media	T-111

Työn valvoja

Petri Vuorimaa

Työn ohjaaja

Timo Engblom

Diplomityössä on tutkittu World Wide Webissä (WWW) tiedonsiirtoon käytettävää Hypertext Transfer Protocol (HTTP) tiedonsiirtoprotokollaa ja toteutettu välityspalvelin, jonka avulla voidaan kerätä mahdollisimman tarkasti tietoja välityspalvelimen kautta liikkuvasta tietoliikenteestä. Kerättyä materiaalia varten on luotu tiedonanalysoinnin rajapinta, jonka avulla eri asiantuntijat voivat toteuttaa omia sovelluksiaan tiedon analysointiin. Rajapinnan käyttäjästä tarjoamaa tietoa voidaan käyttää muun muassa sovelluksien virheiden etsimiseen, raportointiin ja tilastointiin sekä käytettävyystesteihin.

Välityspalvelin on suunniteltu tukemaan käyttäjän tietojärjestelmässä tekemien tapahtumien tallentamista. Käyttäjien tietojen erittely kerätystä tiedosta tapahtuu evästeiden sekä käyttäjän koneen IP-osoitteen perusteella. Palvelinta käytetään myös estämään selaimia ja välityspalvelimiin toteutettuja välimuisteja tallentamasta kopiota selatusta sivusta. Jos välimuisti palauttaa selaimelle oman version haetusta resurssista, ei voida olla varmoja, että kaikki tiedot WWW-sivuston tai -järjestelmän käytöstä saadaan tallennettua tiedon keräystä hoitavalle välityspalvelimelle. Työssä toteutettu välityspalvelin tukee HTTP-protokollan versiota 1.1.

Toteutettua välityspalvelinta käytettiin osana projektinhallintajärjestelmän käytettävyystestausta keräämään tietoja käyttäjän palvelimelle lähettämistä tiedoista. Tämän lisäksi tiedonanalysoinnin rajapinta auttoi testien jälkeisiä testeissä kerättyjen materiaalien indeksointeja. Analysoinnin helpottamiseksi toteutettiin sovellus, jossa voidaan liittää yhteen käytettävyystesteissä syntyvää materiaalia kuten videoita, muistiinpanoja sekä välityspalvelimen tallentamia tietoja. Tämän lisäksi sivuston käytöstä on mahdollista luoda graafeja, jotta voidaan visuaalisesti tarkastella käyttäjän liikkumista sivustolla.

Avainsanat

HTTP, WWW, välityspalvelin, proxy, selain, tiedon kerääminen, välimuisti, eväste, käyttäjän yksilöiminen, käytettävyys, HTTP/1.1

ESIPUHE

Tämä diplomityö sai alkunsa Salomaa Yhtiöiden kanssa sovitusta heidän käytössään olevan projektinhallintajärjestelmän käytettävyystutkimuksesta. Haluankin kiittää Timo Engblomia diplomityön ohjaajana toimimisesta, Veikko Lehtolaa tuesta ja rakentavista keskusteluista sekä Leena Paanasta Salomaa Yhtiöistä tämän diplomityön mahdollistamisesta. Haluan kiittää mahdollisuudesta kehittää itseäni kuluneiden vuosien varrella Salomaa Yhtiöiden eri tytäryhtiöissä toimiessani.

Käytettävyystutkimuksen jälkeen diplomityön eteneminen pysähtyi hetkeksi. Uusia voimia diplomityön loppuun saattamiseen sain PM&RG:n Mervi Rannalta, haluan kiittää häntä kiinnostuksesta diplomityöhöni sekä annetusta palautteesta ja ohjauksesta.

Varsinaisen diplomityöprojektin ulkopuolelta haluan kiittää hyvää ystävääni Jukka Paajasta koko opiskeluajan varrelta. Hän on tukenut minua silloin kun hommat eivät ole menneet kuten tarkoitettu ja kiire on yllättänyt jälleen kerran. Jukan ideat ja asiantuntemus ovat auttaneet paljon tämän diplomityön edistymistä.

Työni valvojaa Petri Vuorimaata kiitän ennen kaikkea positiivisesta suhtautumisesta työhöni, kiinnostuksesta sen etenemistä kohtaan sekä siitä että olet jaksanut olla hyvin kärsivällinen. Kiitokset myös arvokkaista neuvoista ja kommenteista, joita hän on minulle työni aikana antanut.

Kiitän vanhempiani kaikesta siitä tuesta, jonka olen heiltä opiskeluni ja elämäni aikana saanut, ilman teitä en olisi nyt tässä.

Lopuksi tahdon vielä kiittää rakasta vaimoani ja tytärtäni siitä, että olette jaksaneet silloin kun iskä on lähtenyt taas kerran aamulla töihin ja palannut vasta ilta myöhään lämpimään kotiimme.

Esipuhe 3
Taulukkoluettelo 6
Kuvaluettelo 7
Käytetty sanasto ja niiden käännökset 8
1. Johdanto 9
2. Tutkimuksen tavoitteet 12
2.1. Tutkimuksen rajaukset 13
3. Taustaa 14
3.1. HTTP-protokolla 15
3.1.1. HTTP:n perusteet 17
3.1.2. HTTP-palvelupyyntö ja -vastaus 19
3.2. Web-liikenteen välittäjät 21
3.2.1. IBM WBI 23
3.3. Tapahtumien keräysmenetelmät 24
3.3.1. Lokit 25
3.3.1.1. Lokitiedostomuodot 27
3.3.1.2. Lokien analysointityökalujen tuottama tieto käytöstä 31
3.3.1.3. Lokitiedostojen käytön yhteenveto 32
3.3.2. Eväste sekä JavaScript -pohjaiset tiedonkeräimet 33
3.3.2.1. Platform for Privacy Preferences (P3P) 36
3.3.2.2. Evästeiden ja JavaScript:n avulla saatava tieto 36
3.3.2.3. Evästeiden ja JavaScript:n käytön yhteenveto 38
3.3.3. Referer-viittaukset 39
3.3.3.1. Referer-viittauksien avulla saatava tieto käytöstä 40
3.3.3.2. Referer-viittauksien yhteenveto 42
3.3.4. Asiakaspäässä tehtävä tiedonkeruu 42
3.3.4.1. Asiakaspäässä tehtävällä tiedonkeruulla saatavat tiedot 43
3.3.4.2. Asiakaspäässä tehtävän tiedonkeruun yhteenveto 43

4. HTTP-välityspalvelin 45
4.1. HTTP-välityspalvelimen toimintalogiikka ja arkkitehtuuri 47
4.1.1. Sisällön muodostaja 48
4.1.2. Sisällön muokkaaja 49
4.1.2.1. HTTP-protokollan vaatimukset välityspalvelimille 51
4.1.2.2. Välimuistien käytön estäminen HTTP-otsikoiden avulla 52
4.1.2.3. Käyttäjästä kerättävien tietojen kerääminen 54
4.1.3. Sisällön tallentaja 57
4.2. Tapahtumien kerääminen 58
4.3. Tiedonanalysoinnin rajapinta 60
4.3.1. Käytettävyystestit 63
4.3.2. Sovellustestaus 63
4.3.3. Tilastointi ja raportointi 64
5. Projektihallintajärjestelmän käytettävyysarviointi 65
5.1. Testitilanne 65
5.2. Testitehtävät 66
5.3. Välityspalvelimen käyttö ja kerättävän tiedon konfigurointi 67
5.4. Testitilanteiden analysointi 69
5.5. Tiedon analysointisovellus 70
6. Johtopäätökset 73
6.1. Kokemukset järjestelmän käytöstä 73
6.1.1. Kerätyn tiedon rajaaminen 73
6.1.2. Sivulla vietetyn ajan selvittäminen 74
6.1.3. Tiedon analysointi 75
6.1.4. Välityspalvelimen käyttöönotto 76
6.1.5. Muut huomiot 76
6.2. Saatiinko oikea data kerättyä? 77
6.3. Kehityskohteita 78
6.3.1. Tapahtumien koostaminen useammasta sivupyynnöstä 78
6.3.2. HTTP-viestien muokkaaminen ulkoisissa sovelluksissa 79
6.3.3. HTTP-välityspalvelin HTTP-palvelimeksi 80
7. Lähteet 81
Liite 1: HTTP-protokollan tilakoodit 86

TAULUKKOLUETTELO

Taulukko 1: Käytetty sanasto ja niiden käännökset. 8
Taulukko 2: TCP/IP-malli asiakas-palvelin-mallisessa tietoliikenteessä 16
Taulukko 3: Tietokenttien välittäminen palvelupyynnössä HTTP-palvelimelle. 20
Taulukko 4: NCSA Common Logfile Format:n kuvaus. 27
Taulukko 5: W3C:n Extended Log Format:n kuvaus. 28
Taulukko 6: W3C:n Extended Log Format:n kuvaus. 30
Taulukko 7: Lokitiedoista saatavat tiedot sivupyynnöistä. 31
Taulukko 8: Lokitiedoista saatavat tiedot sivuston linkityksestä. 31
Taulukko 9: Lokitiedoista saatavat muut tiedot. 32
Taulukko 10: Lokitiedostojen käytön yhteenveto. 32
Taulukko 11: Evästeiden sekä JavaScriptin avulla saatavat tiedot käyttäjän koneesta. 37
Taulukko 12: Evästeiden sekä JavaScriptin avulla saatavat tiedot selaintekniikoista. .38
Taulukko 13: Evästeiden ja JavaScriptin käytön yhteenveto. 39
Taulukko 14: Referer-viittauksien avulla saatavat tiedot. 41
Taulukko 15: Referer-viittauksien yhteenveto. 42
Taulukko 16: Asiakaspäässä tehtävän tiedonkeruun avulla saatavat tiedot. 43
Taulukko 17: Asiakaspäässä tehtävän tiedonkeruun yhteenveto. 44
Taulukko 18: Välityspalvelimen käyttämän tietokantataulun kuvaus. 59
Taulukko 19: Tallennettavat tiedot sekä muutokset HTTP-vastaukseen. 68
Taulukko 20: HTTP:n palvelupyynnön vastauksien tilakoodien luokat. 86
Taulukko 21: Ilmoitusluontoiset tilakoodit. 86
Taulukko 22: Palvelupyynnön onnistuessa palautettavat tilakoodit. 86
Taulukko 23: Uudelleenohjauksen yhteydessä palautettava tilakoodit 87
Taulukko 24: Asiakasohjelmiston virheen yhteydessä palautettavat tilakoodit 88
Taulukko 25: Palvelinohjelmiston virheen yhteydessä palautettavat tilakoodit. 89

KUVALUETTELO

Kuva 1: WWW-palvelinten lukumäärä maailmassa (NetCraft 2004 ©). 18
Kuva 2: HTTP-palvelupyyntö. 19
Kuva 3: HTTP-vastaus. 20
Kuva 4: Käyttäjän pakottaminen käyttämään välityspalvelinta. 22
Kuva 5: WBI:n arkkitehtuuri (Kuva © IBM [WBI, kappale "Architecture"]). 23
Kuva 6: Käyttäjän yksilöiminen evästeen avulla. 33
Kuva 7: Tiedon välitys kolmannen osapuolen palvelimelle. 35
Kuva 8: HTTP-välityspalvelimen eri sijoittelumahdollisuudet. 46
Kuva 9: Kokonaisarkkitehtuurin kuvaus. 47
Kuva 10: Sisällön muodostaja -osan kuvaus. 49
Kuva 11: Sisällön muokkaaja -osan kuvaus. 50
Kuva 12: Välimuistin käyttötapahtuma 53
Kuva 13: Sisällön tallentaja -osan kuvaus. 57
Kuva 14: Konfiguraatioeditori. 61
Kuva 15: Analyysitiedostot. 61
Kuva 16: GraphML:n pohjalta yEd Graph Editorilla luotu graafi sivuston käytöstä. 62
Kuva 17: Tapahtumien kirjaaminen tiedon analysointisovellukseen. 70
Kuva 18: Tapahtumien kytkeminen heuristiikkoihin analysointisovelluksessa. 71
Kuva 19: Yhteenveto tiedon analysointisovelluksessa heuristiikoista. 72
Kuva 20: HTTP-palvelimen emulointi 80

KÄYTETTY SANASTO JA NIIDEN KÄÄNNÖKSET

Termi	Englannin kielinen vastine tai selitys
TCP/IP-malli	TCP/IP model
WWW	World Wide Web
URL-osoite	URL-address (Uniform Resource Locator)
Etäkutsu	Remote Procedure Call
MIME	Multipurpose Internet Mail Extension
HTTP-palvelin	HTTP server
Web-liikenteen välittäjät	Web Intermediates
HTTP-yhdyskäytävä	HTTP gateway
HTTP-tunneli	HTTP tunnel
HTTP-välityspalvelin	HTTP proxy
Palomuuri	Firewall
HTTP-pyynnön tyyppi	HTTP request method
HTTP-vastauksen tilakoodi	HTTP response status code
Komentokieli	Script language
Katkeamaton yhteys	Persistent connection
Sovelma	Applet
Käytettävyysarviointi	Usability evaluation
Käytettävyystutkimus	Usability inspection
Käytettävyystestaus	Usability testing
Heuristinen arviointi	Heuristic evaluation
HTTP-resurssi	Kokonainen HTTP-kysely tai HTTP-vastaus sisältäen sekä HTTP-otsikot että sisällön
Eväste	Cookie. HTTP-palvelimen selaimeen tallentama tietue.

Taulukko 1: Käytetty sanasto ja niiden käännökset.

1. JOHDANTO

WWW-pohjaiset (World Wide Web) järjestelmät valtaavat tietotekniikka-alaa päivä päivältä yhä enemmän. Nykyisiä järjestelmiä muokataan niin, että niiden käyttöliittymä voidaan vaihtaa WWW-pohjaisiksi. Monet yrityksille elintärkeät sovellukset, kuten verkkopankki-, laskutus-, dokumentinhallinta- sekä sähköpostijärjestelmät ovat jo nyt käytettävissä WWW-käyttöliittymillä.

Selkeänä etuna WWW-pohjaisissa järjestelmissä on ylläpidon helppous. Järjestelmästä on aina käytössä sama versio kaikilla ja järjestelmän ylläpidossa ei jouduta kiinnittämään niin paljon huomiota asiakassovellusten versioeroihin. Hyvin suunnitellun ja toteutetun WWW-pohjaisen järjestelmän uuden version asentaminen ei vaikuta käyttäjään mitenkään muuten kuin uusien ominaisuuksien opettelun kautta.

Tämän diplomityön tarkoituksena on selvittää eri tapoja taltioida WWW:ssä käytettyä HTTP-pohjaista (Hypertext Transfer Protocol) tietoliikennettä ja käyttäjän WWW-pohjaisten järjestelmien käytöstä saatavia muita tietoja sekä myöhemmin analysoida kerättyjä tietoja. Tietoliikenteen seurannalla voidaan esimerkiksi selvittää sovelluksissa olevia ohjelmointivirheitä, tilastoida käyttöä ja tehdä erilaisia raportteja.

Nykyinen HTTP-sovellusten seuranta on pitkälti sovelluskohtaista. Selaimen ja sovelluksen välisen keskustelun tallennus tapahtuu joko HTTP-palvelimen tai sovellukseen kytketyn tiedonkeräimen toimesta. Sovelluskohtaiset ratkaisut eivät ole kuitenkaan välttämättömiä ja kovinkaan yleinen ratkaisu tiedonkeruuongelmaan. HTTP-pohjaisessa tietoliikenteessä kaikki tieto käyttäjän selaimen ja WWW-palvelimen välisessä HTTP-tietoliikenteessä on salaamatonta ja näin tiedon välittäjänä toimivien välityspalvelimien seurattavissa. Lisäksi sovelluskohtaiset ratkaisut ovat mahdottomia tilanteissa, joissa käytetyn järjestelmän lähdekoodia ei voida muuttaa seurantaa varten.

HTTP-tietoliikenteessä sovellukset välittävät tietokentät ja niiden arvot palvelimen sekä käyttäjän koneelle asennetun WWW-selaimen välillä aina samassa muodossa riippumatta siitä miten sovellus on toteutettu. Tästä yhdenmukaisuudesta johtuen on mahdollista toteuttaa tiedonkerääjä sekä tiedon analysointirajapinta, joka ei ole riippuvainen seurannan kohteena olevan järjestelmän toteutuksesta.

Toteutetussa välityspalvelimessa käyttäjän tietoliikenteen seuraaminen pohjautuu suoraan tietoliikenteeseen käytettyyn HTTP-protokollaan ja näin ollen se kykenee tarjoamaan yhdenmukaisen menetelmän tietojen keräämiseen riippumatta testatusta WWW-sovelluksesta. Koska kerätty tieto on samanmuotoista, on myöhemmin tehtävät tiedon analysoinnit mahdollista ulottaa yksittäisistä palveluista jopa useampiin eri palveluihin. Tiedon analysointirajapinta sekä välityspalvelin hoitavat käyttäjien yksilöimisen ja tiedon tarjoamisen jokaisesta välityspalvelimen avulla kerätystä samanmuotoisessa XML-tiedostossa. Näiden XML-tiedostojen analysointiin voidaan käyttää eri käyttötarkoituksiin toteutettuja tiedon analysointisovelluksia. Tässä diplomityössä on toteutettu tiedon analysointisovellus helpottamaan käyttäjätesteissä syntyvän materiaalin indeksointia.

Itse tietojärjestelmiin sekä niiden käyttöön liittyy paljon erilaisia ongelmia, joita voidaan selvittää tutkimalla palvelimen sekä asiakas-ohjelmiston välistä keskustelua. Näitä ongelmia ovat mm. erityyppiset käytettävyysongelmat, sovelluksen toteutukseen liittyvät ongelmat sekä erityyppiset raportit ja tilastot.

Useasti ongelman ratkaiseminen muokkaamalla tutkittavaa järjestelmää on erittäin työlästä ja usein jopa mahdotonta, kun järjestelmän lähdekoodiin ei ole pääsyä. Koska tietoliikenne voidaan ohjata HTTP:n 1.1 [HTTP/1.1] määrittelyn mukaisesti kulkemaan erillisen välityspalvelimen kautta, voidaan kaikki palvelimen ja asiakasohjelmiston välillä liikkuvat tapahtumat saada kirjattua. Varsinaiseksi tutkimusongelmaksi muodostuu näin 1. käyttäjien yksilöiminen tiedon analysointia varten, 2. käyttäjästä, selaimesta, palvelimesta sekä tietoliikenteestä saatavien tietojen selvittäminen, 3. selainten ja välityspalvelimien välimuistien käytön estäminen sekä 4. kerätyn tiedon tarjoaminen eri käyttötarkoituksiin.

Kappaleessa 2 esitellään tutkimuksen tavoitteet ja rajaukset. Tutkimuksen tavoitteita ovat muun muassa toteuttaa tiedon keräykseen soveltuva välityspalvelin, tutkia erityyppisiä tiedonkeruumenetelmiä ja käyttää välityspalvelinta osana käytettävyystestiä.

Kappaleessa 3 on selitetty tämän diplomityön kannalta olennaisia taustatietoja, joihin lukeutuvat erityyppisten tiedonkeruumenetelmien sekä HTTP-tiedonsiirtoprotokollan perusteet.

Kappaleessa 4 esitellään toteutettu välityspalvelin ja tiedonanalysoinnin rajapinta. Tämän lisäksi esitellään menetelmiä yksilöidä käyttäjä kerätystä materiaalista.

Kappaleessa 5 esitellään käytettävyystestit välityspalvelimen näkökulmasta.

Kappaleessa 6 esitellään käytettävyystesteissä tehdyt havainnot ja mahdollisia jatkokehityskohteita.

2. TUTKIMUKSEN TAVOITTEET

Tavoitteena on tuottaa tiedonkeräin, joka kerää asiakasohjelmiston (WWW-selain) ja palvelinohjelmiston (HTTP-palvelin) välisestä tietoliikenteestä mahdollisimman tarkkaa informaatiota myöhempiä analyysejä varten.

Tavoitteina on luoda tutkittavasta WWW-pohjaisesta järjestelmästä riippumaton tapa seurata ja muuttaa järjestelmän sekä asiakasohjelmiston (WWW-selaimen) välillä kulkevaa tietoliikennettä. Järjestelmän ja asiakasohjelmiston välistä tiedonsiirtoa on muutettava välimuistien käytön estämistä varten ja eri käyttäjien yksilöimiseksi kerätystä tiedosta. Tämän lisäksi on tavoitteena luoda rajapinta kerättyyn tietoon niin, että kerättyä tietoa voidaan tehdä analysoida monia eri tarkoituksia varten. Toteutettavan välityspalvelinsovelluksen tulee olla helposti käyttöönotettava tiedonkeräysmekanismi, jota voidaan käyttää erityyppisiin järjestelmien testauksiin kuten käytettävyystesteihin ja sovelluksien virheiden selvitykseen.

Kun testin kohteena olevan järjestelmän testaus on suoritettu ja näin kerätty tieto on tallessa, voi järjestelmän testaaja analysoida kerättyä tietoa tiedonanalysoinnin rajapinnan kautta. Rajapinta tuottaa kerätystä materiaalista tietorakenteen, jota voidaan analysoida erityyppisillä menetelmillä. Tarkoituksena on toteuttaa helppokäyttöinen työkalu, jonka avulla voidaan tallentaa tietoa WWW-pohjaisen järjestelmän käytöstä ymmärtämättä HTTP-tietoliikenneprotokollaa.

Tässä diplomityössä toteutetaan yksi tiedonanalysointisovellus käytettävyystestien käyttöä varten. Tiedon analysointisovelluksessa kiinnitetään huomiota järjestelmässä tapahtuvien tapahtumien ja videomateriaalin indeksointiin mahdollisimman helposti liittymärajapinnan kautta saadun tietorakenteen pohjalta.

Työssä tutkitaan välityspalvelimen soveltumista tiedon keräämiseen ja selvitetään

A. kyettiinkö tarjoamaan riittävän helppokäyttöinen ratkaisu tiedon tallentamiseen ja analysointiin?

B. jouduttiinko rajaamaan kerättävän tiedon määrää tai laatua?

C. tarjoaako HTTP-välityspalvelin lisäetua muihin tiedonkeruumenetelmiin nähden?

2.1. Tutkimuksen rajaukset

Tutkimus rajataan käsittämään tietyn tapahtuman kirjaaminen vain, jos kyseinen tieto välittyy HTTP-protokollan avulla asiakasohjelmiston ja palvelimen välillä. Tutkimuksen ulkopuolelle rajataan selaimessa tapahtuva tiedon laskenta ja käyttöliittymään tapahtuvat dynaamiset muutokset kuten DHTML (Dynamic HyperText Markup Language) ja Microsoftin content-editable ominaisuus. Selainikkunassa tapahtuvia paikallisia tapahtumia ei siis tallenneta. Esimerkki paikallisesta tapahtumasta on tilanne, jossa käyttäjä syöttää lomakkeille tietoja ja selain suorittaa laskentaa käyttäen jotakin selainpohjaista komentokieltä kuten esimerkiksi JavaScript.

Tutkimuksessa kerätään tietoa pelkästään TCP/IP-mallin [IPSUITE] sovelluskerroksessa tapahtuvista tapahtumista, eikä siis esimerkiksi kuljetuskerroksen tietoja. Näin rajataan kerätyn tiedon ulkopuolelle tilanteet, joissa palvelimen ja selaimen välillä liikkuvassa datassa olisi virheitä. Virheetön tiedonsiirto tapahtuu kuljetuskerroksen toimesta.

Rajataan tämän diplomityön osana tehtävä käytettävyysarviointi Salomaa-yhtiöiden projektinhallintajärjestelmästä sekä arvioinnin dokumentointi koskemaan vain käytettävyystestausta ja sitä miten toteutettu HTTP-välityspalvelin tuki itse käytettävyystestejä sekä tiedonanalysoinnin rajapinta myöhemmin tapahtuvaa käytön analysointia. Diplomityössä toteutetaan tiedon analysointisovellus käytettävyystestausta varten. Tiedon analysointisovelluksessa kiinnitetään huomiota järjestelmän tapahtumien ja videomateriaalin indeksointiin mahdollisimman helposti liittymärajapinnan kautta saadun tietorakenteen pohjalta.

3. TAUSTAA

Tässä kappaleessa esitellään HTTP-tiedonsiirtoprotokolla, erityyppiset web-liikenteen välittäjät sekä nykyisiä tiedonkeruumenetelmiä. Nämä esitellään siinä laajuudessa, että HTTP-välityspalvelimen toteutus, toiminnallisuus, rakenne ja ominaisuudet ovat ymmärrettävissä.

Tiedonkeruumenetelmistä esitellään yleisimmät palvelinlokityypit, eväste ja JavaScript-pohjaiset tiedonkeruumenetelmät, referer-viittaukset sekä asiakaspäässä tehtävä tiedonkeruu. Jokaisesta tiedonkeruumenetelmästä esitellään menetelmän avulla saatavat tiedot. Välityspalvelimen toteutusvaiheessa yhdistellään nämä tiedot, jotta välityspalvelimen tiedonkeräin kykenisi keräämään kaiken tarpeellisen tiedon käyttäjästä.

Nykyisistä määrityksistä tämän diplomityön kannalta olennaisin on Hypertext Transfer Protocol (HTTP) -tiedonsiirtoprotokollan uusin versio 1.1 [HTTP/1.1], jota kaikki nykyiset WWW-selaimet käyttävät tiedonsiirtoprotokollana palvelimelle. HTTP-tiedonsiirtoprotokollan tarkka ymmärtäminen on tärkeää koska välityspalvelimen on toteutettava sekä WWW-selaimen että HTTP-palvelimen toiminnallisuus.

Vaikka HTTP:n version 1.1 määritys on ollut valmis jo vuodesta 1999, on WWW-pohjaisiin järjestelmiin siirtyminen kestänyt johtuen mm. HTML:n perustuvan käyttöliittymätekniikan rajallisuudesta sekä HTTP-palvelin- ja sovellusalustojen keskeneräisyydestä.

Samalla kun yhä useammat tietojärjestelmät siirtyvät vähitellen toimimaan WWW-ympäristössä, tulee mahdolliseksi analysoida ja tilastoida järjestelmien toiminnan ongelmia yhtenäisesti. Kun sovellus siirretään käyttämään tiedonsiirtoprotokollana HTTP:tä, mahdollistetaan tiedon kerääminen tiedon analysointia varten. Vaikka sovellukset toimivat usein aikaisemminkin asiakas-palvelin -mallin pohjalta, ei kahden eri tavalla toteutetun tietojärjestelmän käyttöä ole voitu tarkkailla samalla menetelmällä. Eri sovellusten tiedonsiirtoon käyttämät tiedonsiirtoprotokollat kun olivat usein sovelluskohtaisia.

3.1. HTTP-protokolla

HTTP-protokolla on yksi asiakas-palvelin arkkitehtuurin tyyppiesimerkki. Tietoliikenne WWW:ssä pohjautuu juuri HTTP-protokollaan. WWW koostuu hypermediadokumenteista, jotka on tallennettu HTTP-palvelimille. Jokaisella WWW:ssä olevalla resurssilla kuten HTML-sivuilla, kuvilla, Flash-elokuvilla on sen yksilöivä URL-osoite (Uniform Resource Locator). URL-osoitteet ovat HTTP-palvelimille osoitettaessa muotoa:

```
http://www.esim.fi:80/polku/index.html
```

HTTP kuvaa käytettävää tiedonsiirtoprotokollaa. Osoitteessa "www.esim.fi" tarkoittaa palvelimen verkkotunnusta ja "80" TCP porttia verkkotunnuksen osoittamalla palvelimella. "polku" ja "index.html" kuvaavat resurssin sijaintia palvelimella.

HTTP-protokollassa WWW-selain kuten Mozilla tai Microsoft Internet Explorer tekee palvelupyynnön palvelimelle, joka palauttaa vastauksena haetun resurssin. Protokolla itsessään on tilaton, eli palvelin ei pidä kirjaa edellisistä latauksista [INTERNETWORKING s. 530]. Paljolti tästä johtuen HTTP tukee välityspalvelimien ja välimuistien käyttöä, koska välillä olevat web-liikenteen välittäjät voivat aina olettaa tietävänsä kaiken tarpeellisen tiedon, jota vaaditaan resurssin palauttamiseen WWW-selaimelle. Tämä ei olisi mahdollista, jos palvelin tallentaisi tilatietoja, jotka vaikuttaisivat HTTP-vastaukseen. Koska HTTP ei ole reitittävä protokolla, ei se itse määritä reittiä selaimen ja palvelimen välillä. Tällöin voisi käydä niin, että web-liikenteen välittäjää ei käytettäisi tietyissä tilanteissa. Seuraavalla sivulatauksella ei web-liikenteen välittäjä tietäisi kaikkia tarvittavia tilatietoja.

HTTP-protokolla kuuluu TCP/IP-mallin sovelluskerrokseen [INTERNETWORKING, s. 184]. TCP/IP-mallin eri kerrokset yhdessä kuvaavat mallin tietoliikenneverkolle. TCP/IP-malli koostuu viidestä eri kerroksesta, jotka on listattu alla olevassa taulukossa. Taulukon jokaisella rivillä on esitelty myös HTTP-tietoliikenteessä yleisimmin käytetty kyseisen kerroksen protokolla.

Käyttäjän tietokone		**HTTP-palvelin**
1. Sovelluskerros: HTTP (WWW-selain: mm. Mozilla, IE)		1. Sovelluskerros: HTTP (HTTP-palvelin: mm. Apache, IIS)
↓↑		↑↓
2. Kuljetuskerros: TCP		2. Kuljetuskerros: TCP
↓↑		↑↓
3. Verkkokerros: IP		3. Verkkokerros: IP
↓↑		↑↓
4. Linkkikerros: esimerkiksi Ethernet		4. Linkkikerros: esimerkiksi Ethernet
↓↑		↑↓
5. Fyysinen kerros: esimerkiksi T1		

Taulukko 2: TCP/IP-malli asiakas-palvelin-mallisessa tietoliikenteessä.

Taulukossa käydään läpi hyvin tavallinen asiakas-palvelin -arkkitehtuuripohjainen resurssin lataus palvelimelta. Ensin käyttäjä kirjoittaa WWW-selaimeensa URL-osoitteen. Toisena vaiheena selain muodostaa HTTP-palvelupyyntöviestin ja muodostaa yhteyden HTTP-palvelimeen. Kolmanneksi HTTP-palvelin käsittelee HTTP-palvelupyynnön (Kuva 2: HTTP-palvelupyyntö) ja neljännessä vaiheessa vastaa siihen HTTP-vastausviestillä (Kuva 3: HTTP-vastaus).

Käytännössä sovelluskerrokset eivät suoraan neuvottele keskenään vaan keskustelu hajotetaan useammaksi pienemmäksi paketiksi sovelluskerroksen alla olevien TCP/IP kerrosten toimesta. Palvelimella paketit kootaan yhdeksi HTTP-resurssiksi, joka annetaan HTTP-palvelimen käsiteltäväksi. Tietoliikenne tapahtuu siis käyttäen kaikkia kerroksia. Kuljetus-, verkko- ja peruskerrokset hoitavat virheettömän tiedonsiirron sovelluskerroksen sovellusten välillä.

3.1.1. HTTP:n perusteet

HTTP on toteutettu käyttäen etäkutsujen kaltaista rajapintaa, jossa selaimen ja palvelimen väliset viestit välitetään käyttäen ASCII-pohjaisina (American Standard Code for Information Interchange) selkokielisinä viesteinä [BUILDINGSECURE, s. 171]. Käytännössä HTTP-palvelimen ja WWW-selaimen välinen tietoliikenne tapahtuu pelkästään näiden HTTP-viestien avulla.

HTTP-viestit koostuvat otsikko-osasta ja varsinaisesta viesti-osasta. HTTP-otsikot sisältää tiedot HTTP-viestin sisällöstä ja muita yhteyden kannalta olennaisia tietoja. Viestiosa sisältää pyydetystä resurssista riippuen esimerkiksi kuvan tai sivun lähdekoodin. Viestiosassa käytetään sähköposteista tuttua MIME-koodausta [BUILDINGSECURE, s. 171], jonka avulla viestiosassa on mahdollista käyttää esimerkiksi binäärimuotoista materiaalia. Tärkeää on kuitenkin huomata, että HTTP 1.1 ei ole täysin MIME-yhteensopiva vaan MIME:n määrityksestä [MIME] on poikettu, jotta muun muassa binääritiedostojen siirtoa on voitu optimoida [HTTP/1.1, kappale 19.4].

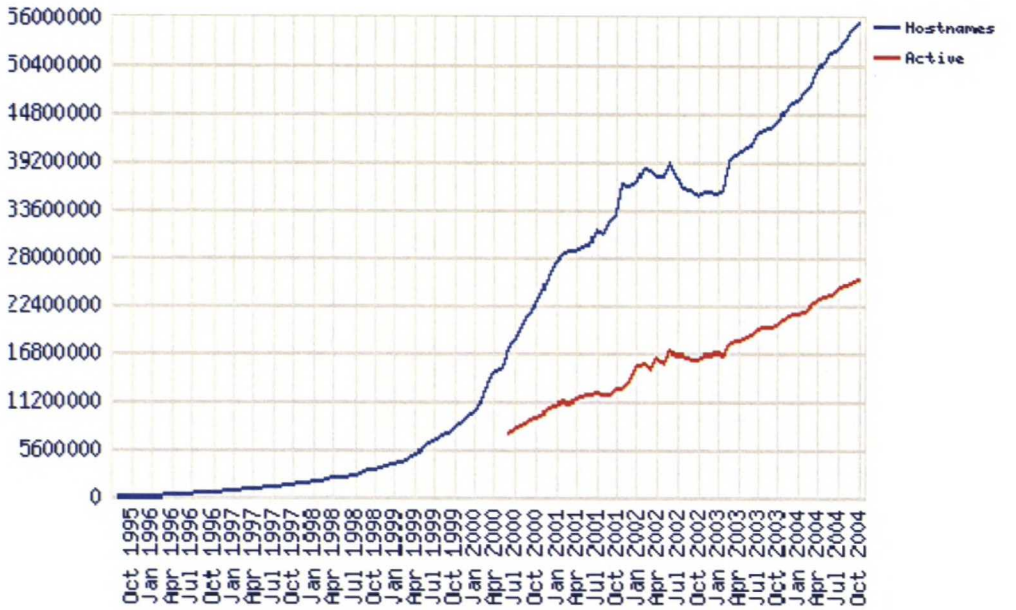
Alla on esitetty yksinkertainen esimerkki HTTP-pyynnöstä, jolla palvelimelta noudetaan HTML-tiedosto WWW-selaimelle. Muita tietoja ei palvelimelle tarvitse välittää.

```
1: GET /uusisivu/index.html HTTP/1.1
2: Host: www.esim.fi
3:
```

Rivillä 1 määritetään haettavan resurssin URL-osoite (Uniform Resource Locators). URL-osoitteen avulla HTTP-palvelin päättää, minkä resurssin palauttaa HTTP-vastausviestissä asiakkaalle.

Rivillä 2 määritetään HTTP/1.1-määrityksen mukaisesti palvelimen Host-osoite. Käytännössä Host on palvelimen verkkotunnus.

Rivi 3 on tyhjä. Tällöin selain tietää vastaanottaneensa kaikki otsikot.

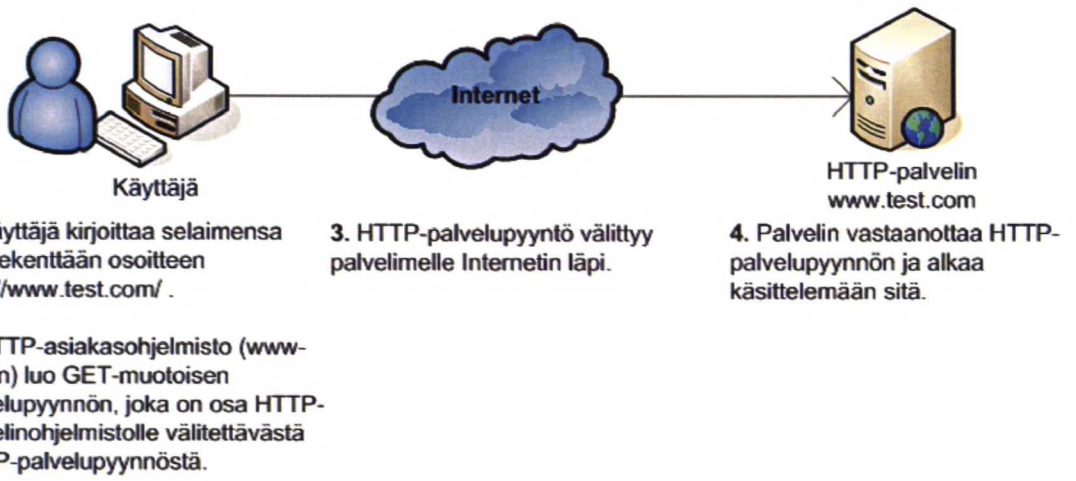


Kuva 1: WWW-palvelinten lukumäärä maailmassa (NetCraft 2004 ©).

HTTP-protokollan version 1.1 oli vastattava huikeaan kasvuun HTTP-palvelinten sekä verkkotunnusten määrässä (kts. kuva 1). Tämän vaatimuksen vuoksi muun muassa `Host`-osoitteen lisättiin jokaiseen HTTP-viestiin. Vanhan määrityksen mukaan yhdessä portissa yhdessä IP-osoitteessa voi toimia vain yksi palvelin. HTTP/1.1-protokollan määritys julkaistiin kesäkuussa 1999. Tätä ennen oli käytössä noin 5.6 miljoonaa HTTP-palvelinta, jotka varasivat jokainen oman IP-osoitteensa. Osittain uuden määrityksen avustuksella palvelinten lukumäärä kasvoi kesäkuuhun 2000 mennessä kolminkertaiseksi lähes 18 miljoonaan. Jos `Host`-kentässä ei ole määritetty porttia, olettaa WWW-selain kohdeportiksi 80 [BUILDINGSECURE, s. 170].

Johtuen HTTP-pohjaisen tietoliikenteen kasvusta oli myös kyettävä optimoimaan jokaista sivulatausta. Tästä syystä esiteltiin katkeamaton yhteys (persistent connection) palvelimen ja selaimen välille. HTTP-protokollan edellisessä versiossa 1.0 jokainen HTTP-pyyntö tehtiin käyttäen erillistä TCP-yhteyttä. Versioon 1.1 lisättiin mahdollisuus käyttää samaa TCP-yhteyttä. Näin kaikki yhden sivun lataukseen liittyvät resurssit kuten kuvat, flash-elokuvat yms. saadaan ladattua niin haluttaessa yhtä TCP-yhteyttä pitkin. [INTERNETWORKING, sivu 532]

3.1.2. HTTP-palvelupyyntö ja -vastaus



Kuva 2: HTTP-palvelupyyntö.

Palvelupyyntö koostuu HTTP-otsikosta sekä HTTP-viestistä. HTTP-otsikko jakautuu sekä palvelupyynnössä että vastauksessa kahteen osaan: ensimmäisellä rivillä olevaan pyynnön tai vastauksen tyypin ilmoittamaan otsikkoon sekä muihin HTTP-otsikoihin.

Palvelupyyntötyyppejä on yhteensä kahdeksan erilaista, joista yleisimmässä käytössä ovat GET, POST, HEAD.

GET-palvelupyyntöä käytetään yleisimmin. GET palvelupyynnön avulla voidaan ladata palvelimelta tietty HTTP-resurssi. Palvelin palauttaa sekä HTTP-otsikot että HTTP-viestin. Esimerkiksi WWW-sivun lataaminen tapahtuu yleisimmin käyttäen juuri GET-palvelupyyntöä.

HTTP tarjoaa menetelmän siirtää tietokenttiä ja niiden arvoja selaimen ja palvelimen välillä. Tietokentät ja niiden arvot siirretään käyttäen joko POST- tai GET-muotoista palvelupyyntöä. POST-palvelupyynnössä tiedot välitetään palvelupyynnön viestiosassa. GET-palvelupyynnössä tiedot välitetään URL-osoitteessa. Kun tiedot välitetään viestiosassa, voidaan palvelimelle siirtää varmemmin huomattavasti suurempia tietomääriä. HTTP:n versio 1.1 ei rajaa URL-osoitteen pituutta, mutta vanhemmat selaimet eivät tue yli 256 merkin pituisia osoitteita [HTTP/1.1, kappale 3.2.1].

Taulukkoon 3 on kerätty HTTP-pyynnöt välitettäessä tietokenttiä palvelimelle:

GET-muotoinen palvelupyyntö
`GET /index.html?muuttujanimi=arvo1&muuttujanimi2=arvo2 HTTP/1.1` `Host: www.esim.fi`
POST-muotoinen palvelupyyntö

`POST /index.html HTTP/1.1`
`Host: www.esim.fi`
 	
`muuttujanimi=arvo1&muuttujanimi2=arvo2`

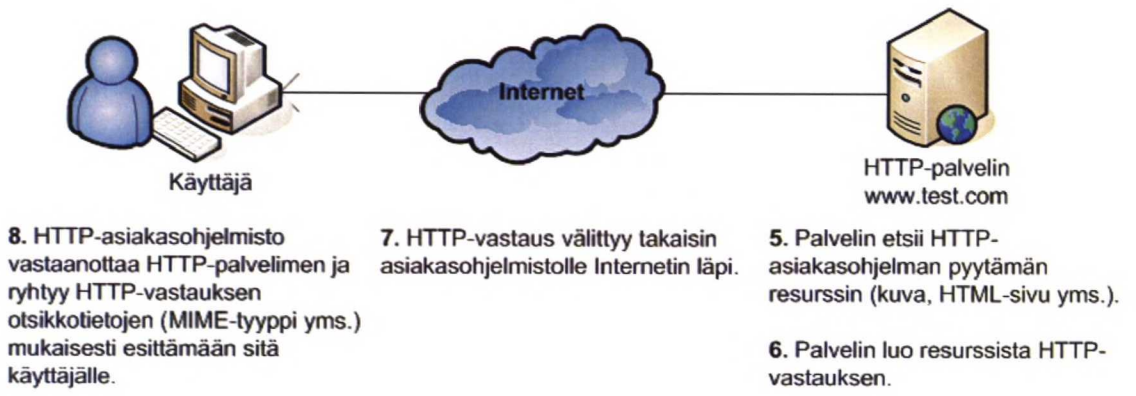
Taulukko 3: Tietokenttien välittäminen palvelupyynnössä HTTP-palvelimelle.

HEAD-palvelupyyntöä käytetään kun halutaan saada selville tietoja tietystä HTTP-resurssista. HTTP-palvelin palauttaa HTTP-vastauksessa ainoastaan HTTP-otsikot ilman viestiosaa. Näin saadaan selville mm. onko kyseinen resurssi vielä olemassa.

Varsinaiset otsikot HTTP-pyynnössä ovat aina muotoa

```
Otsikko: arvo
Referer: http://www.google.fi/search?hl=fi&q=hakusana&lr=
```

Kuva 3 havainnollistaa HTTP-vastauksen kulun palvelimelta käyttäjän selaimelle.



Kuva 3: HTTP-vastaus.

HTTP-palvelupyynnön vastaus muodostuu myöskin otsikoista sekä viesti-osasta. Ensimmäisellä rivillä ilmoitetaan HTTP-vastauksen muoto ja tilakoodi. Kun selain on tehnyt GET-muotoisen kyselyn palvelimelle ja hakee sieltä resurssia joka löytyy palvelimelta, palauttaa palvelin WWW-selaimelle tilakoodin "200 OK" otsikolla:

```
HTTP/1.1 200 OK
```

Jos toivottua resurssia ei olisi löytynyt olisi HTTP-palvelin palauttanut virhekoodin 404 Not Found otsikolla:

```
HTTP/1.1 404 Not Found
```

HTTP:n version 1.1 määrityksessä esitellyt tilakoodit löytyvät tämän dokumentin lopusta (kts. liite 1). HTTP-otsikot löytyvät HTTP:n version 1.1 määrityksestä osoitteesta http://www.w3.org/Protocols/rfc2616/rfc2616.html kappaleesta 14.

Jos haettu resurssi löytyy palvelimelta, se palautetaan HTTP-vastauksen viestiosassa. Alla on esimerkki kokonaisesta HTTP-vastauksesta. Ensimmäisellä rivillä on HTTP-palvelimen ilmoittama tilakoodi. Tilakoodin pohjalta WWW-selain päättelee, miten HTTP-vastauksessa olevien HTTP-otsikoiden pohjalta toimitaan. Tilakoodi "200 OK" ilmoittaa selaimelle haetun resurssin löytyneen. Tämän lisäksi vastauksessa on dokumentin päiväys ja viestiosan tyyppi. Viestinosan tyypin pohjalta selain päättelee, onko viestiosassa oleva sisältö esimerkiksi kuva vai HTML-tiedosto.

```
1:  HTTP/1.1 200 OK
2:  Date: Tue, 31 Aug 2004 09:00:42 GMT
3:  Content-Type: text/html
4:
5:  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN">
6:  <HTML>
7:  <HEAD><TITLE>Testisivu</TITLE></HEAD>
8:  <BODY>
9:  Testisivun sisältöä...
10: </BODY>
11: </HTML>
```

3.2. Web-liikenteen välittäjät

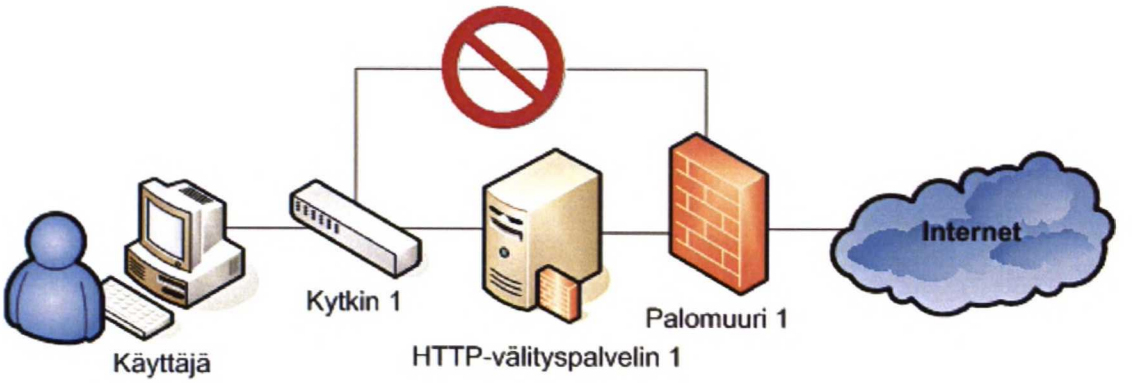
HTTP-tietoliikenteessä voi kohdepalvelimen ja käyttäjän www-selaimen välissä olla erilaisia web-liikenteen välittäjiä. Välittäjien tarkoitus vaihtelee niiden tyypin mukaan. Välittäjiksi luetaan HTTP/1.1-määrittelyn [HTTP/1.1] mukaan HTTP-välityspalvelin, -yhdyskäytävä sekä -tunneli.

"HTTP-välityspalvelin toimii sekä palvelimena että asiakasohjelmistona, jotta se voi tehdä HTTP-palvelupyyntöjä asiakasohjelmiston puolesta. Läpinäkyvällä välityspalvelimella tarkoitetaan välityspalvelinta joka ei muuta palvelupyyntöjä eikä vastauksia.

HTTP-yhdyskäytävä toimii HTTP-pyyntöjen ja HTTP-vastausten välittäjänä asiakasohjelmiston ja palvelimen välillä. Toisin kuin välityspalvelin yhdyskäytävä vastaanottaa HTTP-pyynnöt kuin se olisi kohde HTTP-palvelin. Asiakasohjelmisto ei välttämättä tiedä liikennöivänsä yhdyskäytävän kautta.

HTTP-tunnelilla tarkoitetaan sidosta kahden erillisen yhteyden välillä. HTTP-tunneli lakkaa olemasta kun yhteys alkuperäisen palvelimen ja WWW-selaimen välillä katkaistaan." [HTTP/1.1]

Välityspalvelinten käyttö on tärkeä osa Webin arkkitehtuuria, koska ne tarjoavat välimuisteina toimiessaan tavan vähentää vasteaikoja ja palvelimille syntyviä ruuhkia [INTERNETWORKING, s. 535]. Välityspalvelin on kuitenkin asennettava selaimeen käyttöön eikä näin tarjoa ratkaisua kuin erityyppisten organisaatioiden, kuten koulujen, yritysten sekä internet-palvelutarjoajien käyttöön. Selaimiin on toteutettu tätä varten tekniikka, jonka avulla selain voi ladata automaattisesti välityspalvelinasetukset organisaation määrittämästä osoitteesta. Näin voidaan tehdä välityspalvelinasetukset suoraan useille selaimille. Kuitenkin tämä automaattisten asetusten haun osoitekin on käyttäjän itse määritettävä selaimeen. Selainpuolen tekniikoilla ei voida koskaan varmistua, että käyttäjät käyttävät välityspalvelinta selatessaan WWW-sivuja.

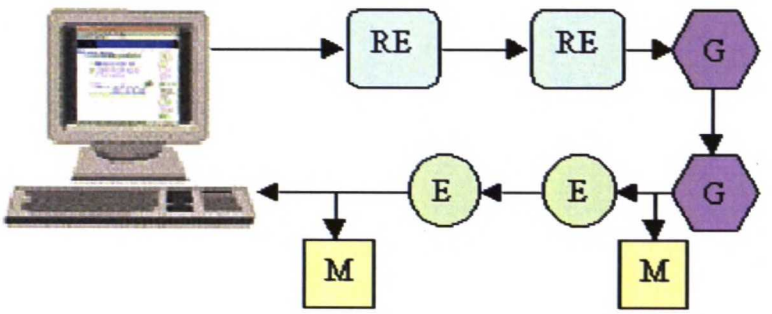


Kuva 4: Käyttäjän pakottaminen käyttämään välityspalvelinta.

Jos halutaan varmistua välityspalvelimen käytöstä, on ainoa mahdollisuus estää HTTP-liikenne sisäverkosta ulkoverkkoon muilta tietokoneilta kuin välityspalvelimelta. Yksi mahdollinen menetelmä välityspalvelimen käyttöönottoon useissa koneissa kerralla ilman käyttäjien vaatimia lisätoimia esitellään kappaleessa 6.3.

3.2.1. IBM WBI

IBM on toteuttanut ohjelmoitavan välityspalvelimen nimeltään WBI (Web Intermediates). WBI tarjoaa sovelluskehittäjälle valmiin välityspalvelimen, joka hoitaa tiedon käsittelyn asiakasohjelmistolle ja selaimelle päin käyttäen HTTP-protokollan versiota 1.0 [WBI, kappale "Architecture"].



Kuva 5: WBI:n arkkitehtuuri (Kuva © IBM [WBI, kappale "Architecture"]).

WBI:n arkkitehtuuri pohjautuu neljään eri komponenttiin, jotka huolehtivat tiedon käsittelystä selaimen ja palvelimen välillä. Nämä komponentit ovat tiedon muodostaja (generator = G), palvelupyynnön muokkaaja (request editor = RE), vastauksen muokkaaja (editor = E) ja tiedon tarkkailija (monitor = M).

Kun selain tekee pyynnön, sen ottaa käsittelyyn ensin palvelupyynnön muokkaaja. Palvelupyynnön muokkaaja välittää tiedon mahdollisine muutoksineen kohdepalvelimelle tai muulle tiedon muodostajalle. Tämän jälkeen tiedon tarkkailija ja tiedon muokkaaja saavat kopion palvelupyynnön vastauksesta. Tiedon muokkaaja tekee vastaukseen tarvittavat muutokset ja välittää tiedon selaimelle sekä tiedon tarkkailijalle.

WBI ei tarjoa mekanismia muuhun kuin HTTP-tietoliikenteen kautta kulkevan tiedon tallentamiseen. Näin tiedot mm. käyttäjän selaimen ominaisuuksista ja sivulla vietetystä ajasta eivät tallennu järjestelmään. Ongelmana voi myös pitää HTTP:n vanhan 1.0 version käyttöä toteutuksessa, jolloin uusilla HTTP 1.1 -pohjaisilla palvelimilla olevat resurssit eivät ole aina käytettävissä. Ongelma ilmenee silloin, kun samassa IP-osoitteessa on useampia verkkotunnuksia ja selain ei välitä palvelimelle HTTP:n version 1.1 määrityksessä olevaa Host HTTP-otsikkoa.

WBI tarjoaa kuitenkin sovelluksen arkkitehtuuriltaan varsin järkevän ja käytännössä ainoan mahdollisen menetelmän tiedon muokkaamiseen sekä tallentamiseen matkalla selaimelta palvelimelle ja takaisin. Jos tietoa ei käsiteltäisi kaikissa WBI:n arkkitehtuurikuvauksessa määritetyissä paikoissa, ei välityspalvelin voisi tallentaa ja muokata kaikkia tarvittavia tietoja palvelupyynnöistä ja -vastauksista.

3.3. Tapahtumien keräysmenetelmät

HTTP-tietoliikenne sisältää paljon erilaisia tapahtumia, jotka kerätään erityyppisillä keräysmenetelmillä. Eri tiedonkeruumenetelmät eivät ole suoraan toisiinsa verrattavia koska ne kaikki ovat perusluonteeltaan hyvin erilaisia. Tässä kappaleessa selvitetään eri tiedonkeruumenetelmien vahvuudet ja heikkoudet sekä mitä tietoa ne tarjoavat käyttäjästä ja hänen tekemistään toiminnoista.

Pääpiirteittäin tiedonkeruumenetelmät voidaan jakaa neljään eri kategoriaan.

1. Lokipohjaiset tiedonkeräimet: jokainen HTTP-palvelinohjelmisto tallentaa lokia sivupyynnöistä ja näiden pohjalta voidaan tehdä tilastoja sivuston käytöstä.

2. Eväste sekä JavaScript -pohjaiset tiedonkeräimet: evästeiden avulla voidaan tarkasti yksilöidä käyttäjä ja tarkkailla miten hän palaa sivustolle myöhemmin. Evästeet mahdollistavat käyttäjien pitkäaikaisen seurannan. JavaScript:n avulla saadaan selaimesta selville monia tärkeitä lisätietoja.

3. Referer-viittauksiin pohjautuvat tiedonkeräimet: käyttäjän saapuessa sivustolle toiselta sivustolta, saadaan tästä tieto WWW-selaimen välittämän Referer-otsikon kautta. Referer-viittauksia käytetään yleisesti sivustojen tilastojen keräämiseen. Sivuille asetetaan viittaus kuvatiedostoon ulkoiselle tilastopalvelimelle. Kun selain hakee kuvan palvelimelta, välittää se samalla Referer HTTP-otsikon sivusta jolle kuva ladataan.

4. Asiakaspäässä tehtävään tiedonkeruuhun pohjautuvat tiedonkeräimet: käyttäjän koneella ajettavat tiedonkeruuohjelmat sekä erityyppiset käytettävyystestit ovat asiakaspään tiedonkeruuta.

Seuraavissa kappaleissa on esitelty tiedonkeruumenetelmät tarkemmin ja lueteltu tapahtumat ja tiedot, joita kunkin keräysmenetelmän avulla voidaan tietoliikenteestä tallentaa. Tietoja käytetään hyödyksi, kun suunnitellaan tietoja, jotka toteutettavan välityspalvelimen tulisi kerätä.

3.3.1. Lokit

Lokipohjainen tiedon tallennus tapahtuu yleensä suoraan HTTP-palvelimen toimesta. Yleisimmät lokin tallennusmuodot ovat NCSA:n Common Logfile Format (CLF), Apachen käyttämä Extended Log Format (ELF) sekä Microsoft IIS:n käyttämä W3C:n määrittelemä Extensible Log Format. Apache palvelinsovellusta käytetään 67,92%:ssa ja Microsoftin IIS palvelinsovellusta 21,09%:ssa WWW:n HTTP-palvelimista (NetCraft Web Server Survey, http://www.netcraft.com/survey). Lokitiedon kuten muunkin käyttäjästä tallennettavan tiedon keräämiseen tulee aina tapahtua käyttäjän suostumuksella, eikä sitä tule tehdä salassa [USABILITYENGINEERING, s.171].

Lokien käsittely on melko suoraviivaista ohjelmointia. Yleisenä periaatteena kaikissa esiteltävissä lokitiedostomuodoissa on se, että yhdelle riville on tallennettu toivotut tiedot koko tapahtumasta eli sekä HTTP-pyynnöstä että -vastauksesta ja kenttien erottamiseen käytetään jotain ennalta määritetty merkkiä, yleensä välilyöntiä. Eri lokimuotojen tukeminen riippuu lähinnä lokien analysointiohjelman arkkitehtuurista eikä niinkään eri lokitiedostomuotojen monimutkaisuudesta.

Lokien analysointiin on olemassa monia varsin kehittyneitä ja helppokäyttöisiä analysointiohjelmia. Yhtenä varsin monipuolisena ja samalla ilmaisena sovelluksena voidaan mainita AWStats -nimisen lokitiedoston analysointisovelluksen, joka toimii GNU GPL lisenssin alla. Lisätietoja sovelluksesta löytyy URL-osoitteesta: http://awstats.sourceforge.net/ .

Lokit mahdollistavat käyttäjän seurannan lähinnä vierailtujen sivujen kautta. Ne eivät yksilöi käyttäjää kovinkaan tarkasti, koska käyttäjästä tallennetaan yksilöivänä tietona pelkkä IP-osoite. Uudemmat lokitiedostomuodot, kuten ELF, tukevat evästeiden tallennusta. Evästeitä voidaan käyttää käyttäjien tarkkaan yksilöintiin. Lokitiedostot toimivat vain lähinnä tietovarastona eivätkä itsessään tarjoa tapaa lähettää selaimelle yksilöivää evästettä. Evästeiden toimittaminen selaimelle on tehtävä aina HTTP-palvelimen, asiakaspäässä ajettavan ohjelman kuten Java-sovelma tai komentokielen kuten JavaScript toimesta. JavaScriptin sekä evästeiden käyttö tiedon keräämiseen ja tallentamiseen esitellään kappaleessa 3.3.2.

Käytännössä lokimenetelmän käytön ongelmana on sen tuottaman aineiston määrä. Kerätty tieto tulee esikäsitellä ennen kuin siitä aletaan tehdä jatkotutkimuksia. Kun kerätty tieto analysoidaan kokonaan koneellisesti, ei esikäsittely ole niin tärkeää kuin ihmisen analysoidessa tietoja [USABILITYENGINEERING, s.171]. Kun tieto on ihmisen käsiteltävänä, on tärkeää, että vain tarpeelliset tiedot ovat esillä.

Kappaleessa 3.3.1.1 esitellään kolme yleisintä lokitiedostomuotoa sekä niiden tarjoamat tiedot käyttäjästä ja sivulatauksista. Seuraavassa kappaleessa 3.3.1.2 esitellään lokitiedostojen analysointityökaluja ja kerrotaan mitä tietoa ne tarjoavat eri tiedostomuotojen keräämän tiedon pohjalta.

3.3.1.1. Lokitiedostomuodot

NCSA Common Logfile Format (CLF)

NCSA:n käyttämä lokitiedostomuoto [NCSALOKI] on otettu käyttöön jo WWW:n ajanlaskun alkupuolella vuonna 1995. Siksi useimmat lokitiedostojen analysointisovellukset tukevat sitä [LOKIANALYSOINTI].

CLF:n mukaisessa lokipohjaisen tiedon tallennuksessa käytetään seuraavaa formaattia riviä kohti lokitiedostossa:

```
remotehost rfc931 authuser [date] "request" status bytes
```

Yhdelle riville lokitiedostoon tallennettavat tiedot on eritelty toisistaan välilyönnillä.

Taulukossa 4 selitetään eri kenttien merkitykset.

Kenttä	Selite
remotehost	WWW-selaimen WWW-palvelimelle näkyvä DNS/IP-osoite (Domain Name System). Voi olla eri kuin varsinainen IP-osoite johtuen erilaisista verkkojen reititystekniikoista kuten NAT (Network Address Translation).
rfc931	RFC (Request for Comments) 931 mukainen tietyn TCP-yhteyden käyttäjän tunniste.
authuser	Käyttäjätunnus jonka avulla käyttäjä on kirjautunut käyttäen http-autentikointia.
date	Päiväys ja kellonaika.
request	WWW-palvelimelle tehty HTTP-pyyntö.
status	WWW-selaimelle palautettu HTTP-vastauskoodi. "200 OK" palautetaan jos resurssin haku on onnistunut.
bytes	WWW-palvelimen palauttaman resurssin sisällön pituus ilman otsikoita.

Taulukko 4: NCSA Common Logfile Format:n kuvaus.

W3C:n Extended Log Format (ELF)

W3C:n lokitiedostomuoto ei ole niinkään oma formaattinsa vaan se täydentää NCSA:n CLF:ää kolmella kentällä [LOKIMUODOT]. Etuna CLF-lokitiedostomuotoon verrattuna on ELF:n tuki evästeille. Evästeiden avulla voidaan käyttäjän sivulataukset yksilöidä lokitiedoissa. W3C:n mukaisessa lokipohjaisen tiedon tallennuksessa käytetään seuraavaa formaattia riviä kohti lokitiedostossa:

```
host rfc931 username date:time request statuscode bytes
referrer user_agent cookie
```

Yhdelle riville lokitiedostoon tallennettavat tiedot on eritelty toisistaan välilyönnillä.

Taulukossa 5 selitetään eri kenttien merkitykset.

Kenttä	Selite
host	WWW-selaimen WWW-palvelimelle näkyvä DNS/IP-osoite. Voi olla eri kuin varsinainen IP-osoite johtuen erilaisista verkkojen reititystekniikoista kuten NAT (Network Address Translation).
rfc931	RFC (Request for Comments) 931 mukainen tietyn TCP-yhteyden käyttäjän tunniste.
username	Käyttäjätunnus, jonka avulla käyttäjä on kirjautunut käyttäen HTTP-autentikointia.
date:time	Päiväys ja kellonaika.
request	WWW-palvelimelle tehty HTTP-pyyntö.
statuscode	WWW-selaimelle palautettu HTTP-vastauskoodi. "200 OK" palautetaan, jos resurssin haku on onnistunut.
bytes	WWW-palvelimen palauttaman resurssin sisällön pituus ilman otsikoita.
referrer	Edellisen sivun URL-osoite.
user_agent	Käyttäjän WWW-selain.
cookie	WWW-selaimen palvelimelle lähettämät evästeet.

Taulukko 5: W3C:n Extended Log Format:n kuvaus.

Extensible Log Format

Extensible Log Format on Microsoft IIS:n käyttämä lokitiedostomuoto, joka tarjoaa käyttäjälleen mahdollisuuden itse määritellä lokitiedostoon kirjoitettavat kentät.

Extensible Log Format:n pohjaisessa lokitiedon tallennuksessa käytetään seuraavaa formaattia:

```
#Software: Microsoft Internet Information Server 6.0
#Version: 1.0
#Date: 1998-11-19 22:48:39
#Fields: date time c-ip cs-username s-ip cs-method cs-uri-
stem cs-uri-query sc-status sc-bytes cs-bytes time-taken
cs-version cs(User-Agent) cs(Cookie) cs(Referrer)

1998-11-19 22:48:39 206.175.82.5 - 208.201.133.173 GET
/global/images/navlineboards.gif - 200 540 324 157
HTTP/1.0 Mozilla/4.0+(compatible;+MSIE+4.01;+Windows+95)
USERID=CustomerA;+IMPID=01234 http://www.loganalyzer.net
```

Ensimmäiset kolme riviä kertovat lokin tiedostomuodon version, lokin aloituskellonajan sekä ohjelmiston jonka tiedon tallentamiseen se on tarkoitettu.

Neljännellä rivillä (Fields) on kuvattu mitä tietoja lokiin on tarkoitus tallentaa. Extensible Log Format:n etu on siinä, että käyttäjä voi itse määrittää siihen tallennettavat tiedot, ilman että tiedot keräävään sovellukseen jouduttaisiin tekemään muutoksia.

Taulukossa 6 on esitelty kaikki mahdolliset kentät, joita voidaan Extensible Log Format:n avulla lokiin tallentaa. Edellisen sivun esimerkissä ei ole niistä käytetty kuin osaa. Kentän käyttöönotto tapahtuisi lisäämällä kenttä lokitiedoston #Fields: riville ja tekemällä HTTP-palvelimen vaatimat toiminnot lokitiedostomuodon muutoksien yhteydessä kuten esimerkiksi HTTP-palvelimen uudelleenkäynnistys.

Kenttä	Selite
date	Tapahtuman päiväys.
time	Tapahtuman kellonaika.
c-ip	Selaimen IP-osoite.
cs-username	Autentikoitunut käyttäjätunnus, jos ei autentikointia tehty "–".
s-sitename	Internet palvelun nimi.
s-computername	Palvelimen nimi, jossa loki-tapahtuma tapahtui.
s-ip	Palvelimen IP-osoite, jossa loki-tapahtuma tapahtui.
s-port	Palvelimen portin osoite, johon WWW-selain oli yhteydessä.
cs-method	Selaimen tekemän HTTP-pyynnön muoto.
cs-uri-stem	Resurssin URI-osoite (Uniform Resource Identifier).
cs-uri-query	Selaimen palvelulle tekemä query eli kysymysmerkin jälkeinen osa URL-osoitetta.
sc-status	Tapahtuman tilakoodi. HTTP:ssä onnistuneen tapahtuman tilakoodi on "200 OK".
sc-win32-status	Tapahtuman tilakoodi Microsoft Windowsin käyttöön, tilakoodit voivat poiketa HTTP:n tilakoodeista.
sc-bytes	Selaimen vastaanottamien tavujen määrä.
cs-bytes	Palvelimen vastaanottamien tavujen määrä.
time-taken	Palvelimen HTTP-pyynnön vastauksen tuottamiseen kulunut aika.
cs-version	HTTP:n versionumero, jota WWW-selain käyttää. Versio on joko 1.0 tai 1.1.
cs-host	HTTP-otsikon "host" arvo, eli käytännössä palvelimen verkkotunnus.
cs(User-Agent)	Käyttäjän WWW-selaimen tunniste.
cs(Cookie)	Sivulla käytetyt evästeet.
cs(Referer)	Edellisen sivun osoite.

Taulukko 6: W3C:n Extended Log Format:n kuvaus.

3.3.1.2. Lokien analysointityökalujen tuottama tieto käytöstä

Lokien analysointityökalut tuottavat kerätyn lokitiedon pohjalta erilaisia raportteja ja tilastoja. Koska lokien tiedostomuodot poikkeavat toisistaan hyvin vähän ja niiden muoto on pysynyt vakiona jo pitkään, ovat lokien analysointityökalut hyvin valmiita ja viimeisteltyjä tuotteita.

Tässä kappaleessa esitellään eri analysointityökalujen lokien pohjalta luomat raporttityypit eli käytännössä ne tiedot, joita lokitiedon pohjalta voidaan johtaa sivuston käytöstä. Tiedot esitellään taulukoissa 7 ja 8 omina riveinään. Jokaisesta tiedosta kerrotaan mihin lokitiedoston tietoon kyseenomainen tieto pohjaa. Taulukossa 7 esitellään tiedot sivupyynnöistä ja taulukossa 8 tiedot sivuston linkityksestä.

Tieto	Mihin pohjautuu
Sivupyynnöt	
Kävijöiden lukumäärä	Sivustolla käyneiden kävijöiden lukumäärä. Käyttäjät erotellaan IP-osoitteen perusteella.
Sivujen lataukset	Niiden rivien lukumäärä, joissa URL-osoite ilman query-osaa on sama.
Sivulataukset hakemistoittain	Poistetaan haetun sivun osoitteesta mahdollinen tiedostonimi sekä query-osa.
Sivut, joista sivustolta lähdetään pois	Eritellään käyttäjien sivupyynnöt ryhmiksi yksilöivän IP-osoitteen tai evästeiden pohjalta ja katsotaan viimeinen sivulataus.
Päivän ruuhkahuiput	Tilastoimalla sivuhaut ja jaottelemalla ne tapahtuma-ajankohdan mukaan päivälle.
Eri viikonpäiville jakautunut sivuston käyttö	Tilastoimalla sivuhaut ja jaottelemalla ne tapahtuma-ajankohdan mukaan viikolle.
Sivustolla käyneet hakukoneet	Tarkastellaan `User-Agent` otsikkoa ja verrataan sitä hakukoneiden lähettämiin selain nimiin.
Virheelliset pyynnöt	HTTP-vastauksen tilakoodi. `200 OK` on normaali. `404 Not Found` on tieto siitä, ettei haettua resurssia löytynyt.

Taulukko 7: Lokitiedoista saatavat tiedot sivupyynnöistä.

Tieto	Mihin pohjautuu
Ulkopuoliset lähteet ja sivun linkitys muualta	
Sivu, johon sivustolle saavutaan	Listataan sivut joiden `Referer` -otsikko ei ole ko. sivustolta.
Linkittävät sivut	Otsikossa `Referer` saadaan tieto edellisestä sivusta.
Käytetyt hakukoneet ja hakusanat	Otsikosta `Referer` saadaan edellinen sivu ja sen pohjalta voidaan luoda tieto hakukoneella tehdystä hausta, jos se on GET-muotoinen.

Taulukko 8: Lokitiedoista saatavat tiedot sivuston linkityksestä.

Tieto	Mihin pohjautuu
Muut tiedot	
Käyttäjän IP-osoite / DNS-nimi	TCP/IP-yhteydestä selaimen ja palvelimen välillä saadaan selville vastakkaisen puolen IP-osoite. Nimi IP-osoitteelle saadaan DNS-nimenselvityksen avulla.
Käyttöjärjestelmä	Käyttöjärjestelmä saadaan selville selaimen lähettämästä `User-Agent` otsikosta.
Sivun lähetyspakkausmuoto	Käytetty tiedon pakkausmuoto saadaan selville lukemalla otsikon `Content-encoding` arvo.
Selainversio	Selaimen versio saadaan selville selaimen lähettämästä `User-Agent` otsikosta.
Käynnin pituus	Käyttäjän ensimmäisen ja viimeisen sivulatauksen kellonajan erotus.
Kirjautunut käyttäjä	Selaimen WWW-sivulle lähettämän `Authorization` -otsikon BASE64 dekoodaamalla saadaan selville käyttäjätunnus ja salasana kaksoispisteellä eroteltuna.
Selain	Selain saadaan selville selaimen lähettämästä `User-Agent` otsikosta.
Tiedostokoko	Tiedostokoko saadaan selville otsikosta `Content-length` tai palvelimen muuten ilmoittamasta tiedosta.
Tiedostomuoto	Tiedostotyyppi saadaan selville joko haetusta URL-osoitteesta tai otsikosta `Content-type`.
Suosikkeihin lisääminen	Tieto saadaan selville tarkkailemalla palvelimen juurihakemistossa olevaa favicon.ico tiedostoa, joka ladataan vain kun sivu lisätään selaimen suosikkeihin.
Käyttäjän maa	WWW-palvelimen näkemän IP-osoitteen sisältämän IP-osoiteavaruuden omistaman yrityksen toimimaa. Käyttäjän käyttääessä välityspalvelinta, nähdään välityspalvelimen maa.

Taulukko 9: Lokitiedoista saatavat muut tiedot.

3.3.1.3. Lokitiedostojen käytön yhteenveto

Lokitiedostot tarjoavat mutkattoman menetelmän pienien sivustojen käytön ja käytön ajankohtien analysointiin. Lokitiedostot eivät kerro muusta kuin puhtaasta HTTP-tietoliikenteestä, se ei kerro tarkasti kauan tietyllä sivulla vietettyä aikaa tai esimerkiksi tarkempia tietoja käyttäjän selaimesta. Tätä varten on käytettävä muita menetelmiä.

Hyödyt	Haitat
Lokitiedostojen käyttöönotto vaivatonta ja lokitiedostomuodot melko yleisiä.	Rajallinen määrä tieto mahdollista kerätä.
Ei vaadi käyttäjän koneelle mitään muutoksia. Täysin selainriippumaton.	Ei tarjoa käyttäjän yksilöintiin ratkaisua. Vaatii muiden tapahtumien keräysmenetelmien käyttöä yksilöintiin.
Lokitiedostojen analysointisovellukset ovat viimeisteltyjä ja kehittyneitä.	Lokien muokkaamismahdollisuudet ovat rajallista.

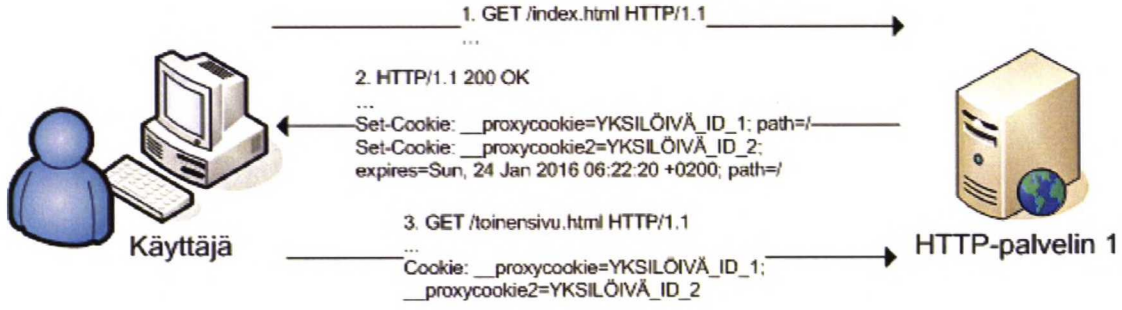
Taulukko 10: Lokitiedostojen käytön yhteenveto.

3.3.2. Eväste sekä JavaScript -pohjaiset tiedonkeräimet

Vaikka JavaScript ja evästeet ovat toisistaan täysin erillisiä tekniikoita, voidaan niitä pitää yhdessä yhtenä tiedonkeräysmuotona johtuen niiden tarjoamasta varsin tehokkaasta tiedonkeruuluonteesta. Tunnetuimpia JavaScript / eväste-pohjaiseen tilastointiin käytettyjä sovelluksia on RedSheriff Inc. toteuttama samanniminen tiedon kerääjä. Suomessa mm. Suomen Gallup käyttää sovellusta tiedonkeruuseen asiakkaidensa sivustoilta sekä Alma Media tytäryhtiöineen käyttää sitä omien sivustojensa seurantaan.

Tärkein yksittäinen tekijä, joka nostaa evästeiden sekä JavaScriptin yhdistelmän lokipohjaisen tiedonkeruun yläpuolelle, on mahdollisuus yksilöidä käyttäjä tarkasti. Kun lokipohjaisissa menetelmissä yksittäisten kävijöiden ja käyntien päättely tapahtuu yleisimmin pelkän IP-osoitteen pohjalta, eivät saman palomuurin tai välityspalvelimen takaa tulevat käyttäjät yksilöidy. IP-osoitteen käyttö yksilöintiin on usein mahdotonta niiden käyttäjien kohdalla, jotka saapuvat tarkkailun alla olevalle sivustolle työpaikkojensa tietoverkoista.

Kuvassa 6 on selvitetty perusperiaate miten käyttäjä voidaan yksilöidä kerätystä tiedosta selaimeen tallennettavan yksilöllisen evästeen avulla.



Kuva 6: Käyttäjän yksilöiminen evästeen avulla.

Kohta 1: HTTP-palvelin havaitsee, ettei selain välitä sille yksilöintievästeitä.

Kohta 2: HTTP-palvelin luo uudet yksilölliset avaimet sekä lyhytaikaista (__proxycookie) että pitkäaikaista (__proxycookie2) seurantaa varten ja sisällyttää ne HTTP-vastaukseen.

Kohta 3: Kun käyttäjä lataa seuraavan sivun "toinensivu.html", välittää selain automaattisesti yksilöivät evästeet HTTP-palvelimelle.

Kahden evästeen käyttö on perusteltua tilanteessa, jossa halutaan erikseen saada yksilöityä käyttäjä pitkällä aikavälillä sekä yksittäisten käyntien tarkkuudella. Tällöin selaimelle välitetään seuraavat tiedot:

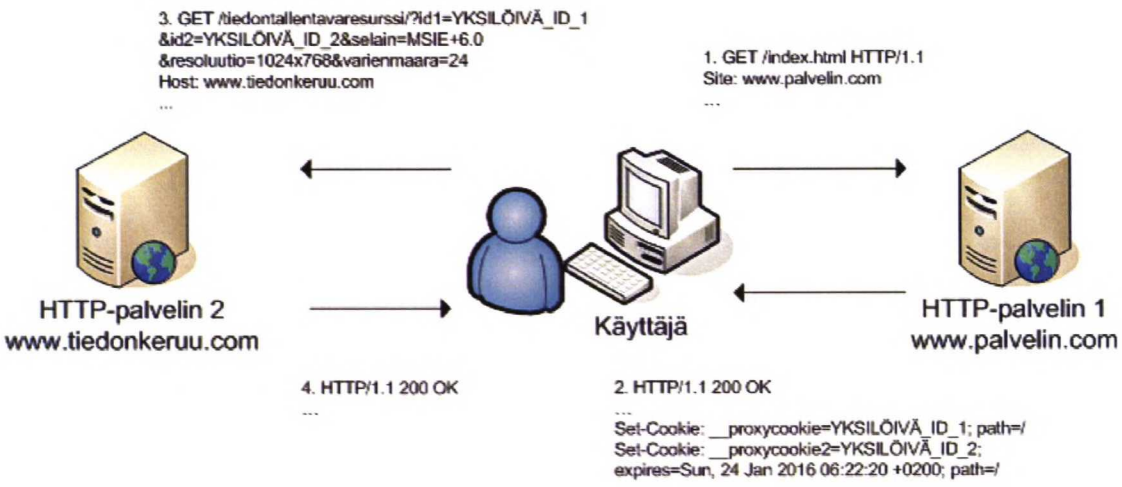
1. Istuntokohtainen eväste (Kuva 6: __proxycookie), joka vanhenee – siis poistetaan käyttäjän koneelta, kun tämä on selaimensa sulkenut. Selain poistaa istuntokohtaisen evästeen, kun selain suljetaan. Tästä johtuen käyttäjän seuraava vierailu sivustolla rekisteröityy uutena käyntinä. Istuntokohtaista evästettä käytetään erittelemään eri kävijöiden käyntejä.

2. "Ikuinen" eväste (Kuva 6: __proxycookie2), jonka vanhenemisaika on määritetty pitkälle tulevaisuuteen. Tämä eväste ei siis poistu selaimesta, kun selain suljetaan. Seuraavalla vierailulla sivustolle selain automaattisesti välittää palvelimelle evästeen. Palvelin tunnistaa käyttäjän ja osaa liittää myöhemmin tämän sivulataukset yhdeksi kokonaisuudeksi. Tulee kuitenkin muistaa, että esimerkiksi selainpäivitykset tai käyttäjän tekemä evästeiden poisto voi tuhota myös ikuiset evästeet. Tällaisessa tilanteessa käyttäjä rekisteröityy tiedonkeräimelle uutena kävijänä seuraavalla kerralla sivustolla vieraillessaan. Ikuisia evästeitä käytetään erittelemään eri kävijöitä.

HTTP-palvelimella evästeet vastaanottava sovellus tallettaa yksilöivän tiedon ja JavaScriptin avulla selvitetyt muut tiedot tietokantaan. Tiedot voidaan välittää toiselle palvelimelle lataamalla toiselta palvelimelta jokin resurssi kuten esimerkiksi pikselin kokoinen kuva, jonka HTTP-pyynnössä välitetään käyttäjän yksilöivä tunniste sekä hänestä esimerkiksi JavaScriptin avulla kerätyt tiedot (kts. kuva 7, kohta 3). HTTP-pyyntö voidaan muodostaa esimerkiksi seuraavasti, jolloin kaikki tieto välitetään suoraan GET-parametreina tiedot tallentavalle palvelimelle.

```
GET http://www.tiedontallentavapalvelin.com/tiedontallenta
varesurssi/?tunnus=testitunnus&id1=yksilöivätieto&id2=yksi
löivätieto2&selain=MSIE+6.0&resoluutio=1024x768
```

Kuvassa 7 on esitetty, miten tieto liikkuu eri palvelinten välillä tiedonkeruun vaiheissa.



Kuva 7: Tiedon välitys kolmannen osapuolen palvelimelle.

Kohta 1: Käyttäjä lataa haluamansa sivun HTTP-palvelimelta 1.

Kohta 2: HTTP-palvelin 1 palauttaa asiakkaalle haetun sivun sekä tallettaa käyttäjälle istuntokohtaisen ja ikuisen evästeen.

Kohta 3: Selaimen palauttaman sivun HTML-koodissa kutsutaan JavaScriptin avulla toiselta palvelimelta resurssia, jolle yksilöivä tieto sekä kerätyt tiedot välitetään esimerkiksi GET-parametrissa.

Kohta 4: HTTP-palvelin 2 tallentaa tiedot tietokantaan ja palauttaa haetun resurssin kuten esimerkiksi pikselin kokoisen kuvan.

Erittäin tärkeää on huomata tapahtumat kohdissa kaksi ja kolme. Jos käyttäjän yksilöinti tehtäisiinkin HTTP-palvelimella 2 kohdassa 3 lisäämällä käyttäjän kohdassa 4 vastaanottamaan HTTP-vastaukseen evästeitä, eivät ne tallentuisi nykyisiin selaimiin eikä käyttäjän yksilöinti ei onnistuisi. Tämä johtuu uusien WWW-selaimien tietoturva-asetuksista, joissa kielletään oletusarvoisesti hyväksymästä evästeitä kolmannelta osapuolelta, jollainen HTTP-palvelin 2 on. Evästeitä hyväksytään vain osoitteista, joihin käyttäjä on itse selaimensa avulla siirtynyt. Ei siis ladatun sivun sisältämien resurssien palvelimilta.

3.3.2.1. Platform for Privacy Preferences (P3P)

Koska kolmannen osapuolen evästeet voivat olla kuitenkin myös hyödyllisiä, on W3C käynnistänyt Platform for Privacy Preferences (P3P) projektin. Käytännössä P3P on kuvaus sivuston tallentamista tiedoista ja siitä mihin tietoja aiotaan käyttää. Jos P3P tiedostossa olevat määritykset eivät ole ristiriidassa selaimen tietoturva-asetuksien kanssa, hyväksytään myös tältä palvelimelta tulevat kolmannen osapuolen evästeet. P3P voidaan kytkeä jokaiseen HTTP-resurssiin joko:

A. Sijoittamalla P3P-määritystiedosto HTTP-palvelimen juureen kansioon "w3c" tiedostonimellä "p3p.xml". Tällöin WWW-selain osaa automaattisesti kaivaa tiedoston, jos ja kun sille ilmenee tarvetta.

B. Käyttämällä HTTP-otsikkoa. Käytetty HTTP-otsikko on "P3P". Otsikon avulla kerrotaan osoite P3P-tiedostoon joka sisältää tiedon siitä mitä kerätyillä tiedoilla tehdään. Otsikko voi myös sisältää tiedot lyhennettynä.

```
P3P: policyref="http://www.tiedonkeruu.com/w3c/p3p.xml"
, CP="NOI DSP COR CURa ADMa DEVa TAIa OUR BUS IND UNI
COM NAV INT"
```

C. Käyttämällä HTML-sivulla LINK-viittausta P3P tiedostoon.

```
<link rel="P3Pv1" href="http://catalog.example.com/P3P/
PolicyReferences.xml">
```

3.3.2.2. Evästeiden ja JavaScript:n avulla saatava tieto

Evästeiden ja JavaScriptin avulla on mahdollista selvittää käyttäjästä hyvin paljon eri tietoja ja yksilöidä käyttäjä hyvin tarkkaan. Käytännössä kaikki tieto, joka JavaScriptin avulla on mahdollista selvittää selaimesta ja käyttäjän koneesta, voidaan kerätä. JavaScript ei sisällä kovinkaan tehokasta menetelmää tiedon tallentamiseen, mutta se tarjoaa lisätietoja käyttäjästä ja tarjoaa mahdollisuuden käyttäjän yksilöimiseen.

Evästeiden ja JavaScript:n avulla saadaan kerättyä kaikki tieto, joka on mahdollista kerätä lokitietojen avulla, kunhan tieto ei pohjaudu pelkästään HTTP-otsikoihin. Näiden tietojen lisäksi on saatavilla tarkempia tietoja käyttäjän tietokoneesta. Näiden käyttäjän koneesta tallennettavien tietojen avulla voidaan useasti selvittää tarkemmin mahdollisia ohjelmistovirheitä sekä ongelmatapauksia. Taulukko 11 ei kata kaikkea JavaScriptin avulla kerättävää tietoa, vaan se sisältää tiedot, joita voidaan olettaa tarvittavat tiedon analysoinnissa myöhemmin.

Tieto	**Mihin JavaScript muuttujaan pohjautuu** IE = Internet Explorer NS = Netscape / Mozilla
Tiedot käyttäjän koneesta	

Käytetty kieli	`navigator.browserLanguage (IE)`
	`navigator.systemLanguage (IE)`
	`navigator.userLanguage (IE)`
	`navigator.language (NS)`

Näyttöala ja resoluutio	`window.innerHeight`
	`window.Width`
	`window.screen.width`
	`window.screen.height`

Käyttöjärjestelmä	`navigator.appVersion (sisältää tiedon käyttöjärjestelmästä)`
Selain	`navigator.appVersion`
Tekstin koko	`document.getElementById("documentid").currentStyle.fontsize;`
Värisyvyys	`window.screen.pixelDepth`

Taulukko 11: Evästeiden sekä JavaScriptin avulla saatavat tiedot käyttäjän koneesta.

Tieto	Mihin JavaScript muuttujaan pohjautuu Internet Explorer = IE Netscape / Mozilla = NS
Tuet eri selaintekniikoille	
Acrobat	navigator.plugins["PDF.PdfCtrl.5"] (IE) navigator.mimeTypes["application/pdf"].enabledPlugin (NS)
Flash	navigator.plugins["Shockwave Flash"] (IE) navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin (NS)
Java	navigator.javaEnabled()
JavaScript	HTML-koodin avulla, muuttamalla x.x <script language="JavaScriptx.x" type="text/javascript">
MediaPlayer	Navigator.plugins["MediaPlayer.MediaPlayer.1"] (IE) navigator.mimeTypes["application/x-mplayer2"].enabledPlugin (NS)
QuicTime	Navigator.plugins["QuickTimeCheckObject.QuickTimeCheck.1"] (IE) navigator.mimeTypes["video/quicktime"].enabledPlugin (NS)
RealPlayer	Navigator.plugins["rmocx.RealPlayer G2 Control.1"] (IE) navigator.mimeTypes["audio/x-pn-realaudio-plugin"].enabledPlugin (NS)
Tiedostojen lähettäminen	Tieto saadaan selville selainversion kautta

Taulukko 12: Evästeiden sekä JavaScriptin avulla saatavat tiedot selaintekniikoista.

3.3.2.3. Evästeiden ja JavaScript:n käytön yhteenveto

Evästeiden ja JavaScriptin käyttö mahdollistaa käyttäjän tarkan yksilöimisen ja tätä kautta tarkentaa myöhempää tiedon analysointia, koska käyttäjän tekemät toiminnot istunnon ajalta voidaan erotella omiksi kokonaisuuksiksi erilleen muiden käyttäjien istunnoista. Pelkällä evästeiden ja JavaScriptin käytöllä ei saada selville tietoja, jotka pohjaavat HTTP- kyselyn tai HTTP-vastauksen otsikoihin. Monet näistä HTTP-otsikoista ovat saatavilla kuitenkin JavaScriptin kautta. JavaScriptin avulla selville saatavia otsikkotietoja ovat edellisen sivun osoite `Referer`-otsikko ja tiedot käytetystä selaimesta `User-agent`-otsikko.

Evästeiden sekä JavaScript:n käyttöönotto on mutkatonta ja nopeaa, yleensä sivustolle pitää lisätä vain muutama rivi JavaScript-koodia, joka osoittaa toisella palvelimella olevalle tiedon tallentajalle.

Taulukkoon 13 on koottu yhteenveto menetelmän hyödyistä ja haitoista.

Hyödyt	Haitat
Evästeiden sekä JavaScript:n käyttöönotto vaivatonta ja ilmaisia tiedonkeräimiä on olemassa monia.	Ei saada selville tietoja HTTP-kyselystä eikä HTTP-vastauksesta. Näin ei tiedetä automaattisesti sitä onko sivu virhesivu vai normaali sivu, jolle JavaScript on asennettu.
Ei vaadi käyttäjän koneelle mitään muutoksia. Vaatii selaimelta JavaScript-tuen.	Tietoa on tarjolla paljon, mitä kannattaa tallentaa?
Ilmaisetkin analysointisovellukset kehittyneitä ja tilastot toimivat usein jopa reaaliaikaisesti.	Käyttäjät suhtautuvat eväste-pohjaiseen tiedonkeruuseen negatiivisesti ja kokevat sen yksityisyyden loukkauksena.
Käyttäjän koneesta voidaan kerätä tietoja JavaScriptin avulla. Näin saadaan taustatietoa esimerkiksi asiakkaiden käyttämien selaimien versioista ja niiden tukemista selaimien lisäosista.	
Yksilöi käyttäjän hyvin tarkasti ja mahdollistaa käyttäjän pitkäaikaisen seuraamisen.	

Taulukko 13: Evästeiden ja JavaScriptin käytön yhteenveto.

3.3.3. Referer-viittaukset

Referer-tieto on yksittäinen HTTP-otsikko edellisen sivun URL-osoitteesta, jonka selain välittää palvelimelle HTTP-pyynnössä. Referer-viittausta ei voida aivan suoraan kutsua omaksi tiedonkeruumenetelmäkseen, koska Referer-tiedothan ovat saatavilla sekä loki-pohjaisessa tiedonkeruussa että JavaScriptin avulla.

Referer-viittaus on selvästi parempi kuin muut yksittäiset kentät HTTP-otsikoissa tai JavaScriptissä. Kun muut kentät koskevat puhtaasti sivua tai tapahtumaa, niin Referer-viittaus kytkee tiedon historiaan jota kautta voidaan mm. selvittää käyttäjän liikkumista sivustolla.

3.3.3.1. Referer-viittauksien avulla saatava tieto käytöstä

Referer-viittaus sisältää siis tiedon siitä mikä on ollut edellisen sivun HTTP-kyselyn URL-osoite. Pyydetyn URL-osoitteen avulla voidaan saada selville paljon hyvin mielenkiintoisia asioita, joita muuten jouduttaisiin selvittämään muiden HTTP-palvelimien ylläpitäjiltä.

Koska referer-viittauksen avulla voidaan sivulataukset kytkeä edellisiin sivulatauksiin, käytetään referer-viittausta hyvin useasti mainonnan kuten bannereiden ja maksettujen hakusanojen tehokkuuden tarkkailussa. Taulukossa 14 on esitelty Referer-viittauksien avulla selville saatavia tietoja. Nämä tiedot voidaan myöhemmin kääntää helpottamaan saavutettujen tietojen löytämistä.

Tieto	Mihin pohjautuu
Tiedot käyttäjän koneesta	
Miten käyttäjä liikkuu sivustolla?	Mille sivulle käyttäjä siirtyy miltäkin sivulta. Tästä voidaan päätellä mahdollisia umpikujia sekä havaita sivuston navigaatioon liittyviä ongelmia. Tutkimalla yhdistelmää yksilöivä tunniste ja edellinen sivu voidaan käyttäjän liikkumisesta sivustolla luoda polku.
Miten käyttäjä poistunut sivustolta?	Kun tutkitaan miten käyttäjän liikkumisesta luotu polku päättyy saadaan selville usein sivuston hyödyllisimmät ja hyödyttömimmät sivut. Tilanteissa, joissa käyttäjä löytää haluamansa tiedon sivulta tai havaitsee ettei tieto ole sinne päinkään, hän mahdollisesti sulkee selaimen tai siirtyy toiselle sivustolle.
Miten ja mihin saapunut sivustolle?	Mitkä ovat ne sivut sivustolla, joihin linkitetään ulkoisilta sivustoilta. Näiden sivujen URL:t tulisi pitää mahdollisimman pysyvinä. Sivut ovat todennäköisesti tärkeimpiä ja niihin tulisi olla tarjolla linkit sivuston etusivulla. Tutkimalla niitä sivuja, joissa Referer-viittaus ei osoita omalle palvelimelle, saadaan selville saapumissivut.
Mitkä sivustot linkittävät sivustolleni?	Sivustojen linkittyminen on nykyisin hyvin tärkeää, jotta sivusto löytyisi eri hakupalveluista. (mm. Google pitää sivustojen keskinäistä linkitystä yhtenä suurimmista tekijöistä, kun sivun sijoitusta hakutuloksissa päätetään). Tämän menetelmän avulla pyritään saamaan selville kuinka yleisessä käytössä sivusto on. Linkitystä voidaan käyttää myös hyödyksi, jotta voidaan hankaloittaa sivuston materiaalien käyttöä ulkoisilla sivustoilla. Jos Referer-viittaus on ulkoiselta palvelimelta, ei kuvatiedostoa lähetetä ollenkaan. Näin estetään oman palvelimen kuvien käyttö muiden sivustojen toimesta.
Millä hakusanoilla sivustolle löydetään?	Selvittämällä Referer-viittauksen kautta saatujen sivujen luonteen voidaan Referer-viittausta analysoida ja saada selville mm. millä hakusanoilla sivusto on löydetty. Tämä tapahtuu analysoimalla hakupalvelinkoneelta saapuvien käyttäjien Referer-osoitteita. Alla esimerkkinä Googlen kautta saapuvien käyttäjien Referer-otsikko. `Referer: http://www.google.fi/search?hl=fi&q=hakusana1+hakusana2&lr=` Tästä saadaan selville, että hakusanat lähetetään q-muuttujassa. Jos referer-viittauksessa olisi em. osoite, saataisiin selville, että sivustolle on löydetty hakemalla hakusanoja *hakusana1* ja *hakusana2*.
Lisäävätkö mainokset sivulatauksia?	Mainosten toiminta ja ihmisten saapuminen mainoksen sisältämältä sivulta saadaan yksinkertaisesti tarkkailemalla Referer-viittausta ja vertaamalla tietoa niihin palveluihin, joihin mainokset on asennettu.
Käyttäjän liikkumisen tarkkaileminen toiselta sivustolta.	Referer-viittaus mahdollistaa tiedon tallentamisen toiselta palvelimelta käsin. Jos tämä kolmannen osapuolen palvelin on toteuttanut P3P-tietoturvaprofiilin, käyttäjää voidaan tarkkailla ulkopuoliselta palvelimelta yhtä tarkkaan kuin palvelimelta, jolta käyttäjä sivua lataa. Puhdas referer-viittauksiin pohjautuva statistiikka kolmannen osapuolen palvelimella ei kerää riittävästi tietoa, koska alkuperäiseen sivulataukseen liittyvä referer-tieto puuttuu. Tästä johtuen kannattaa tässä yhteydessä käyttää JavaScriptiä välittämään alkuperäinen Referer-tieto kolmannen osapuolen palvelimelle.

Taulukko 14: Referer-viittauksien avulla saatavat tiedot.

3.3.3.2. Referer-viittauksien yhteenveto

Referer-viittauksen avulla saadaan selville asioita, joiden perusteella voidaan tehdä tärkeitä johtopäätöksiä liittyen sivuston toiminnallisuuteen sekä kehittämiseen. Eri käyttäjien liikkumien polkujen analysointi ja niistä tehtävät johtopäätökset ovat vaikeita ja vaativat matemaattisia malleja. Ratkaisut eivät ole triviaaleja ja ne ovat useasti sivustokohtaisia.

Hyödyt	Haitat
Luo mahdollisuuden käyttäjän liikkeiden tutkimiseen.	Ei itsenäinen tekniikka, vaatii toimiakseen toisen tiedonkeruumenetelmän, jonka osana referer-viittaus tallennetaan.
Luo mahdollisuuden analysoida asioita, jotka tapahtuvat toisilla sivustoilla.	Käyttäjien polkujen analysointi vaikeaa ja vaatii aikaa löytää tiedosta oikeita tuloksia. Ratkaisut ovat mahdollisesti jopa sivustokohtaisia riippuen toteutustekniikasta.
Ei vaadi käyttäjän koneelle mitään muutoksia. Täysin selainriippumaton.	Ei tarjoa käyttäjän yksilöintiin ratkaisua. Vaatii muiden tapahtumien keräysmenetelmien käyttöä yksilöintiin.

Taulukko 15: Referer-viittauksien yhteenveto.

3.3.4. Asiakaspäässä tehtävä tiedonkeruu

Asiakaspäässä tehtävää tiedonkirjausta voidaan suorittaa hyvin monilla eri menetelmillä. Mahdolliset menetelmät voidaan jakaa karkeasti ammattilaisten tekemiin havainnointeihin sekä käyttäjästä tallennettaviin tietoihin erilaisten apuvälineiden avulla. Apuvälineet voivat olla mm. videokameroita, mikrofoneja sekä esimerkiksi asiakkaan koneelle asennettu kuvaruutukaappari, joka nauhoittaa jokaisen käyttäjän tekemän hiiren liikkeen.

Tyypillisesti käyttäjän koneella tehtävää käytettävyystestausta kutsutaan kenttätestiksi tai -kokeiluksi [USABILITYENGINEERING, s.168]. Siinä käyttäjä käyttää testin kohteena olevaa järjestelmää aidossa ympäristössä. Kenttätesteissä käytetään yleisesti käyttötapahtumaa tallentamaan sekä videokameraa että muistiinpanovälineitä.

WWW-selaimeen asennettavat tiedonkeräimet ja sivun muuttaminen tiedon tallennusta varten ovat esimerkkejä asiakaspään tiedonkeruusta. Yksi tällainen menetelmä on kappaleessa 3.3.2 esitelty JavaScript-komentokieltä käyttävä tiedonkeruumenetelmä.

3.3.4.1. Asiakaspäässä tehtävällä tiedonkeruulla saatavat tiedot

Tässä kappaleessa esitellään niitä tietoja, joita ei muilla tiedonkeruumenetelmillä ole ollut mahdollista saada talteen käyttötapahtumasta. Kun käyttäjä käyttää tietojärjestelmiä, saadaan hänen käytöstään tallennettua kaikki ne tapahtumat joihin käyttäjä on päätynyt oman ajattelunsa pohjalta. Käyttötapahtumasta jää paljon sellaisia asioita tallentamatta, joita ei voida havaita tietokonepohjaisten tiedonkeruumenetelmien toimesta. Esimerkiksi käyttäjän päänsä sisällä tekemä päätösprosessi sekä käyttäjän ilmeet ja ääneen tekemä päättely eivät tallennu tietokonepohjaisilla menetelmillä [USABILITYENGINEERING, s.156]. Käyttäjän ääneen tekemän pohdiskelun analysointi on usein hyvin valaisevaa kun tutkitaan, miten järjestelmän tulisi toimia.

Taulukossa 16 on esitelty menetelmän avulla saatavat tiedot.

Tieto	**Mihin pohjautuu**
Tiedot käyttötilanteesta	
Käyttäjän eleet	Käyttäjän eleitä voidaan hyvin tutkia käyttäjätestien jälkeen mm. kuvatun videokameramateriaalin pohjalta.
Käyttäjän hiiren liikkeet	Käyttäjän hiiren liikkeitä voidaan tallentaa muuttamalla sivun lähdekoodia niin, että sivulle ladataan Java-sovelma, joka tallentaa käyttäjän hiiren liikkeitä palvelimelle talteen. Näin havaitaan miten käyttäjä päätyy valitsemaan tietyn linkin.
Käyttäjän toimien perustelut	Käyttäjää voidaan ohjastaan puhumaan käytettävyystestin aikana ja perustelemaan tekemänsä päätökset. Näitä tietoja analysoimalla voidaan saada selville miksi tietty asia onnistui tai miksi jokin meni pieleen.

Taulukko 16: Asiakaspäässä tehtävän tiedonkeruun avulla saatavat tiedot.

3.3.4.2. Asiakaspäässä tehtävän tiedonkeruun yhteenveto

Käytännössä asiakaspäässä tehtävä tiedonkeruu toimii tilanteissa, joissa testissä mukana olevien käyttäjien lukumäärä on pieni. Tämän lisäksi on varmistuttava, että käyttäjät suostuvat esimerkiksi videoitavaksi. Videoiminen koetaan usein henkilökohtaisemmaksi kuin esimerkiksi lokien käyttö tiedonkeruumenetelmänä. Tiedonkeruu asiakaspäässä kykenee selvittämään sellaisia asioita, joita ei muiden tiedonkeruumenetelmien avulla voida selvittää. Asiakaspäässä tapahtuvaa tiedonkeruuta voidaankin pitää hyvänä osatiedonkeruumenetelmänä muiden tiedonkeruumenetelmien yhteydessä.

Taulukkoon 17 on koottu yhteenveto menetelmän hyödyistä ja haitoista.

Hyödyt	Haitat
Mahdollistaa käyttäjän eleiden sekä turhautumisen kirjaamisen.	Asiakaspäässä kerätty tieto tulee kyetä yhdistämään muiden tiedonkeruumenetelmien tallentamaan tietoon jotenkin. Tämä saattaa olla aikaa vievä vaihe kuten esimerkiksi videon indeksointi.
Käyttäjien valitsemien polkujen analysointi on vaikeaa. Käyttäjän tekemien perustelujen pohjalta tämä on kuitenkin huomattavasti helpompaa.	Toimii kun tiedonkeruumenetelmän kohteena on pieni määrä käyttäjiä. Käyttäjien määrän kasvaessa saattaa muodostua liian tehottomaksi vaihtoehdoksi.

Taulukko 17: Asiakaspäässä tehtävän tiedonkeruun yhteenveto.

4. HTTP-VÄLITYSPALVELIN

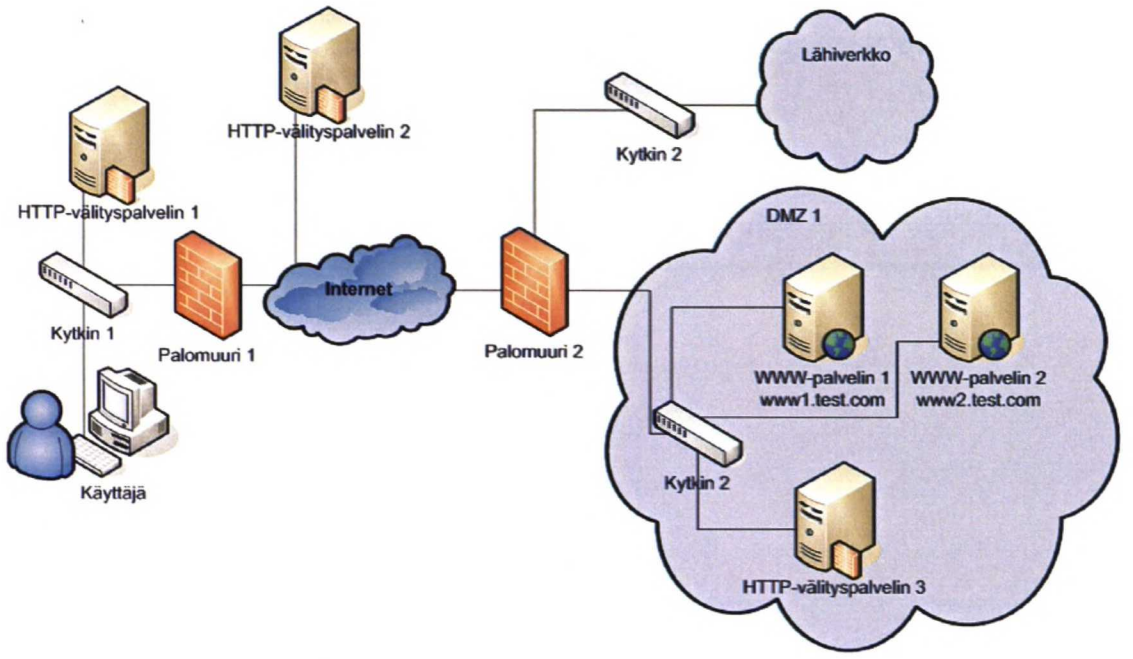
Toteutettava HTTP-välityspalvelin toimii tiedonkerääjänä HTTP-palvelupyynnöissä ja -vastauksissa. Välityspalvelin ei ole HTTP/1.1:n määrityksessä kuvattu läpinäkyvä välityspalvelin vaan muuttaa tietosisältöä sekä palvelupyynnöissä että -vastauksissa.

HTTP-välityspalvelimen käyttö tiedonkeräämiseen verrattuna muihin yleisempiin tiedonkeruumenetelmiin on perusteltua seuraavissa tilanteissa:

1. Muilla tiedonkeruumenetelmillä ei saavuteta niin suurta määrää kerättyä tietoa käyttäjästä. Välityspalvelin kykenee keräämään kaikki tapahtumat kootusti, kun yksittäisten menetelmien tuottamat tiedot voidaan yhdistää jo tiedon keräysvaiheessa.

2. Testin tekijällä ei ole mahdollisuutta muuttaa testin tai tilastoinnin kohteena olevan sivun lähdekoodia. Välityspalvelin tekee itsenäisesti muutokset sivun lähdekoodiin riippumatta sivuston toteutustekniikasta tai siitä kuka sivustoa ylläpitää.

3. Nykyiset tiedonkeruumenetelmät ei toimi johtuen selainten tietoturva-asetuksista. Usein tietoturva-asetukset voidaan kiertää siten, että muokataan HTTP-palvelimen antamaan HTTP-vastausta ja muutetaan selaimelle lähetettävää vastausta siten, että käyttäjän yksilöimiseen tarkoitetut evästeet olisivatkin varsinaiselta palvelimelta.



Kuva 8: HTTP-välityspalvelimen eri sijoittelumahdollisuudet.

HTTP-välityspalvelimen voi sijoittaa tiedonkeruun kannalta kolmeen eri sijaintiin.

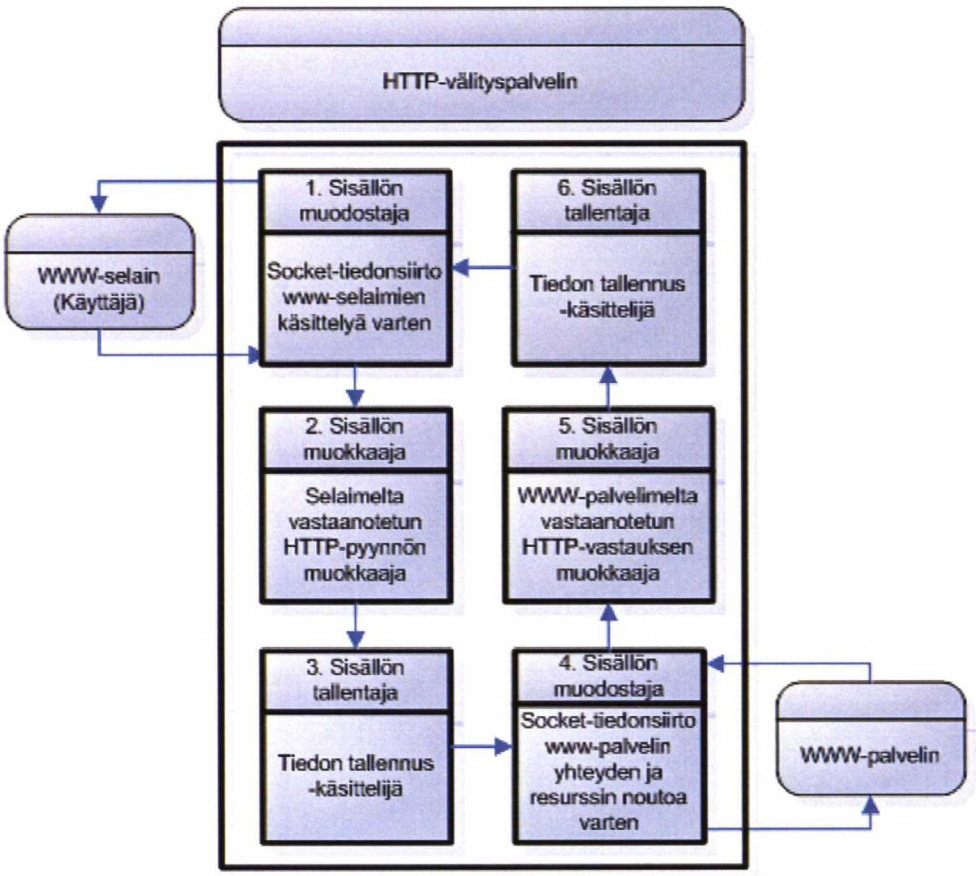
1. HTTP-välityspalvelin 1 kykenee yksilöimään sivuston käytön myös konetasolla koska käyttäjän kone on samassa lähiverkossa välityspalvelimen kanssa. IP-osoitetta voidaan käyttää tilanteessa, jossa halutaan jaotella yrityksen sisältä lähtevää HTTP-liikennettä eri ryhmiin tilastointia varten. Käyttäjäkohtainen ryhmittely tehdään jo mainittujen evästeiden avulla.

2. HTTP-välityspalvelin 2 toimii esimerkiksi tiedonkeräystä hoitavan yrityksen tiloissa sekä käyttäjän että palvelimen lähiverkkojen ulkopuolella. HTTP-välityspalvelin 2 on sijainniltaan tiedonkeruun kannalta kaikkein heikoimmassa asemassa. Sijaintinsa vuoksi se ei tarjoa mitään lisäarvoa tiedonkeruuseen.

3. HTTP-välityspalvelin 3 mahdollistaa käyttäjien liittämisen tiedonkeruuseen ilman, että heidän selaimiaan tarvitsee konfiguroida. Käytännössä sisäverkon nimipalvelimet konfiguroidaan kappaleessa 6.3.3 esitetyllä tavalla. Lyhyesti kuvattuna ulkomaailmasta saapuva DNS-nimikysely tietystä verkkotunnuksesta ohjataan HTTP-välityspalvelimelle. HTTP-välityspalvelin on konfiguroitu käyttämään toista nimipalvelinta, joka tietää WWW-palvelimen oikean IP-osoitteen, jota käyttäen HTTP-välityspalvelin osaa hakea resurssin sieltä.

4.1. HTTP-välityspalvelimen toimintalogiikka ja arkkitehtuuri

Järjestelmän perusarkkitehtuuri on asiakas-palvelin malli, jossa WWW-selain hakee tietyn resurssin WWW-palvelimelta. HTTP-välityspalvelimen arkkitehtuuri perustuu kolmeen osaan, jotka ovat: sisällön muodostaja, -muokkaaja sekä -tallentaja. Nämä toiminnalliset kokonaisuudet vastaavat pitkälti kappaleessa 3.2.1 esiteltyjä IBM:n WBI välityspalvelimen toiminnallisia osia. Käytännössä erilliset palvelupyynnön ja -vastauksen muokkaajat on yhdistetty samaan luokkaan. Tämän lisäksi tiedon tarkkailija on vaihtunut tiedon tallentajaksi, joka ainoastaan tallentaa muiden komponenttien antamat tiedot tietokantaan.



Kuva 9: Kokonaisarkkitehtuurin kuvaus.

Kuvassa 9 kohdassa 1 käyttäjä avaa WWW-selaimestaan tietyn sivuston, välittää WWW-selain välityspalvelimelle HTTP-pyynnön tietystä resurssista WWW-palvelimella. HTTP-välityspalvelimella selaimien pyyntöjä käsittelee sisällön muodostaja. Se hoitaa Socket-pohjaisen tiedonsiirtoyhteyden selaimelle ja vastaanottaa siltä tulevan HTTP-palvelupyynnön (kts. kuva 9, kohta 1).

Kun sisällön muodostaja on vastaanottanut HTTP-pyynnön se välittää sen sisällön muokkaajalle, joka aloittaa HTTP-pyynnön käsittelyn (kts. kuva 9, kohta 2). Sisällön muokkaaja tekee HTTP-pyyntöön tarvittavat muutokset. Muutokset voivat liittyä HTTP-pyynnön otsikoihin tai sisältöön.

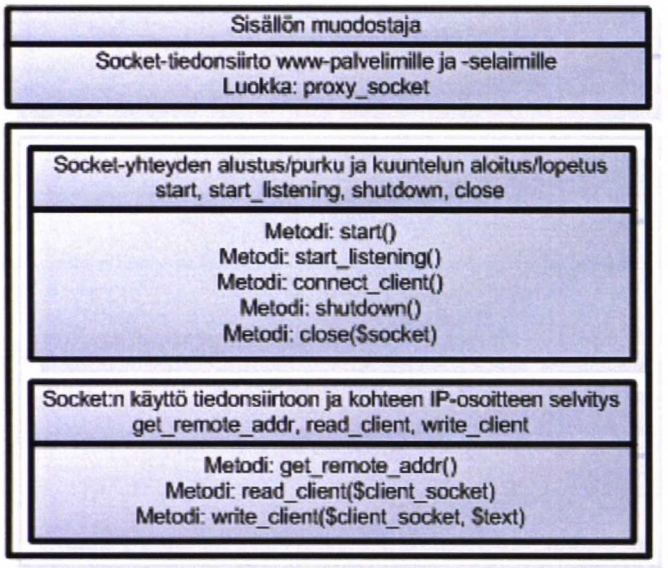
Tarvittavien muutoksien jälkeen sisällön muokkaaja välittää alkuperäisen ja muokatun HTTP-pyynnön tiedon tallentajalle, joka hoitaa tiedon tallentamisen relaatiokantaan (Kuva 9 kohta 3).

Tässä vaiheessa HTTP-pyyntö on vastaanotettu, käsitelty ja tallennettu myöhempää analysointia varten. Nyt HTTP-pyyntö välitetään sisällön muodostajalle, joka hoitaa Socket-pohjaisen tiedonsiirtoyhteyden palvelimelle ja välittää sille muokatun HTTP-pyynnön ja vastaanottaa siltä HTTP-vastauksen (kts. kuva 9, kohta 4).

Sisällön muodostaja välittää HTTP-vastauksen sisällön muokkaajalle, joka tekee HTTP-vastaukseen tarvittavat muutokset (kts. kuva 9, kohta 5) kuten kohdassa 2 ja välittää sekä alkuperäisen että muokatun HTTP-vastauksen sisällön tallentajalle, joka tallentaa tiedon relaatiokantaan. Tämän jälkeen sisällön tallentaja välittää HTTP-resurssin sisällön muodostajalle, joka palauttaa muokatun HTTP-vastauksen selaimelle.

4.1.1. Sisällön muodostaja

Sisällön muodostajaksi kutsutaan niitä menetelmiä järjestelmässä, jotka luovat WWW-selaimelle tietosisällön, kuten esimerkiksi HTML-sivun. Koska järjestelmä toimii välityspalvelimena eikä sisällä välityspalvelimissa yleistä välimuistitoiminnallisuutta on sisällön muodostaja puhtaasti Socket-pohjainen [SOCKET] tiedonsiirtoyhteys selaimen ⇔ välityspalvelimen sekä välityspalvelimen ⇔ palvelimen välillä. Sisällön muodostaja palauttaa siltä pyydetyn resurssin (HTTP-pyynnön tai –vastauksen).

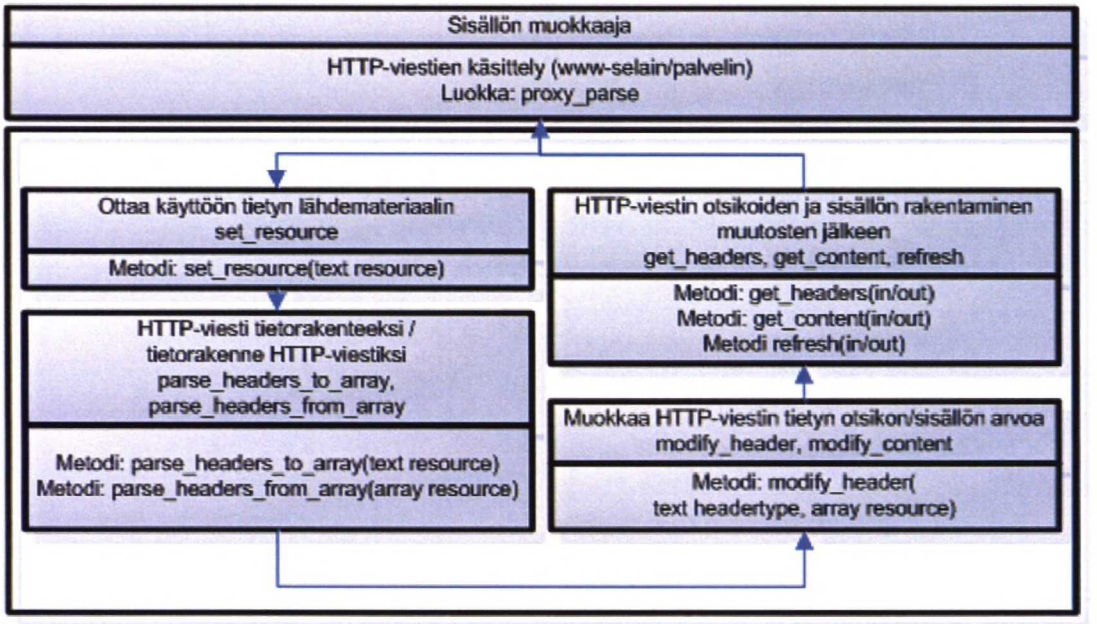


Kuva 10: Sisällön muodostaja -osan kuvaus.

4.1.2. Sisällön muokkaaja

Sisällön muokkaaja muokkaa sisällön muodostajan tuottamaa tietosisältöä. Sen avulla on esimerkiksi mahdollista muovata sivun lähdekoodia tai muokata HTTP-pyynnön otsikoita. Sisällön muokkaajaa käytetään HTTP-välityspalvelimessa lisäämään otsikoita, jotka vaikuttavat siihen ladataanko sivu uudestaan joka kerta sekä lisätietoja varten Java-sovelman latauskoodin. Näin saadaan varmasti jokainen sivulataus tallennettua ja Java-sovelman kautta tallennettua sivulatauksesta lisätietoja, kuten sivulla vietetty aika.

Vaihtamalla sisällön muokkaajan on mahdollista vaikuttaa kerättävään tietoon ja siihen, kuinka tarkkaa ja monipuolista kerätty tietosisältö on. Ohjelmoimalla sisällön muodostaja ja -muokkaaja uudelleen voidaan muuttaa HTTP-välityspalvelin HTTP-palvelimeksi. Lisäksi on mahdollista tarjota HTTP-palvelinta vastaavaa toiminnallisuutta tarjoamalla esimerkiksi tietystä ennalta määritetystä osoitteesta, jonka sisällön muodostaja ja -muokkaaja tietävät, tietoja käyttäjän selaamista kohteista ja tilastoja välityspalvelimen kautta ladatuista sivuista.



Kuva 11: Sisällön muokkaaja -osan kuvaus.

Tiedon muuttaminen WWW-selaimen ja HTTP-palvelimen välisessä tietoliikenteessä on kolmesta syystä pakollista, kun HTTP -välityspalvelinta käytetään tiedonkeruuseen:

1. HTTP-välityspalvelimen on muokattava tiettyjä otsikoita, jotta se noudattaisi W3C:n laatimaa HTTP-protokollan version 1.1 määritystä [HTTP/1.1]. HTTP-protokollan vaatimukset välityspalvelimelle on selvitetty kappaleessa 4.1.2.1.

2. Tiedonkeräyksen kannalta on oleellista, että jokainen sivu ladataan joka kerta HTTP-palvelimelta eikä missään välissä käytetä selaimen tai selaimen ja palvelimen välillä olevia välimuisteja. Jos sivu latautuu välimuistista, ei tieto tapahtumasta taltioidu HTTP-välityspalvelimen tiedonkeräimelle tilanteessa, jossa välimuisti on sijoitettu HTTP-välityspalvelimen ja selaimen välillä. HTTP-protokollan toimintatapa välimuistien käytölle on esitetty kappaleessa 4.1.2.2.

3. Jos tietosisältöä ei muuteta, ei käyttäjän sivulatauksista saada tallennettua muita kuin otsikoihin pohjautuvia tietoja. Tietosisällön muokkaaminen on pakollista, jotta välityspalvelin voi lisätä sivulle tiedonkeräimet, jotka tarjoavat tietoa käyttäjästä sekä hänen koneestaan ja kykenevät yksilöimään käyttäjän.

4.1.2.1. HTTP-protokollan vaatimukset välityspalvelimille

HTTP-protokolla määrittää välityspalvelimille tiettyjä vaatimuksia joita tulee noudattaa, sekä joitain suosituksia, jotta myös vanhemmat tai mahdollisesti virheellisesti toimivat selaimet toimisivat. Jokaisen vaatimuksen ja suosituksen perässä on viittaus HTTP 1.1:n määrityksen kappaleeseen [HTTP/1.1].

Seuraavassa on esitetty HTTP-protokollan vaatimukset välityspalvelimelle:

1. Jos selain tai palvelin välittää HTTP-otsikoissa useamman kuin yhden samantyyppisen HTTP-otsikon, on niiden järjestys pysyttävä samana kun HTTP-viesti välitetään eteenpäin. [4.2 Message Headers]

2. Kun selain välittää kyselyn välityspalvelimelle, on pyydetyn URL-osoitteen oltava resurssin tarkka URL-osoite. Esimerkiksi pelkän polun tai suhteellisen viittauksen välittäminen ei riitä. [5.1.2 Request-URI]

3. HTTP-otsikko `Continue`:n noudattaminen. Jos tiedetään seuraavan välityspalvelimen tukeman HTTP-protokollan olevan 1.0 tai vanhempi, ei HTTP-pyyntöä tule välittää eteenpäin. Selaimelle tulee palauttaa vastaus tilakoodina "`417 Expectation failed`". Jos seuraavan välityspalvelimen tiedetään tukevan HTTP:n versiota 1.1, välitetään viesti aina. [8.2.3 Use of the 100 (Continue) Status]

4. Jos välityspalvelin vastaanottaa HTTP-pyynnön jossa on määritetty HTTP-otsikko `Max-Forwards` ja sen arvo on nolla, ei HTTP-viestiä saa välittää eteenpäin. Jos arvo on suurempi kuin nolla tulee sen arvoa vähentää yhdellä. [9.2 OPTIONS]

5. Välityspalvelin ei saa muuttaa HTTP-otsikoiden: `Content-Location`, `Content-MD5`, `Etag`, `Last-Modified` arvoja. [13.5.2 Non-modifiable Headers]

6. Jokaisessa viestissä on oltava määritettynä `Host` HTTP-otsikko. [14.23 Host]

Seuraavassa on esitetty HTTP-protokollan suositukset välityspalvelimelle:

1. Jos selain tekee HTTP-pyynnön, jossa verkkotunnus ei ole sinällään riittävän tarkka, voi välityspalvelin täydentää sitä. Jos verkkotunnus on tarkka, sitä ei saa muuttaa. Esimerkiksi haku

   ```
   GET http://earthquake/index.html HTTP/1.1
   ```

   voidaan tulkata siis hauksi

   ```
   GET http://earthquake.cs.hut.fi/index.html HTTP/1.1
   ```

   [3.2.2 HTTP URL]

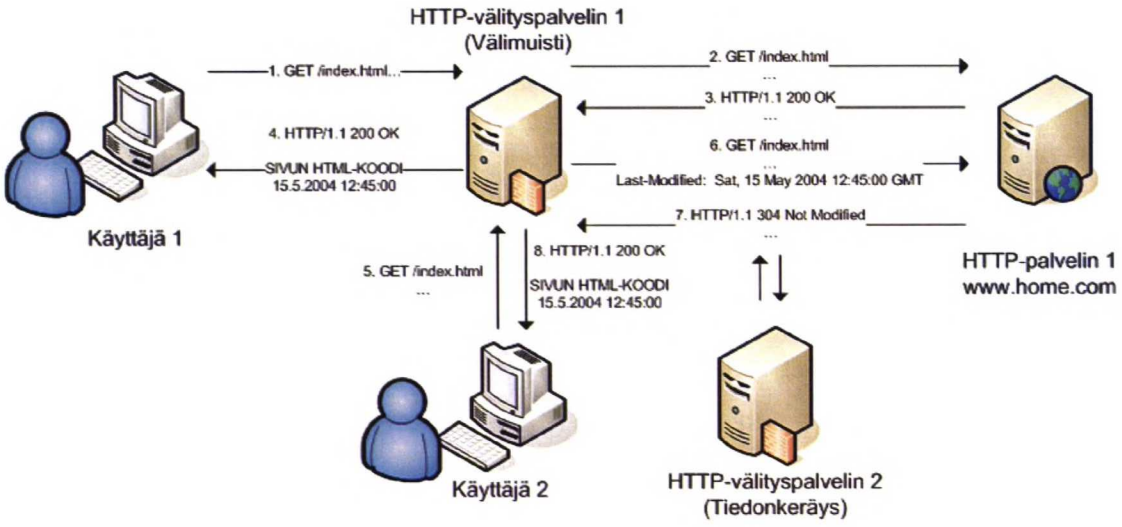
2. Selaimelle ei tule sallia yli kahta samanaikaista yhteyttä palvelimelle eikä välityspalvelimelle yli 2*N (N=yhtäaikaisten käyttäjien lukumäärä). [8.1.4 Practical Considerations]

4.1.2.2. Välimuistien käytön estäminen HTTP-otsikoiden avulla

Välimuistilla tarkoitetaan muualla kuin varsinaisella HTTP-palvelimella olevaa tallennetta selaimen pyytämästä resurssista, joka voidaan lähettää selainohjelmistolle HTTP-palvelimen puolesta. Yleisimpiä välimuistityyppejä ovat erityyppiset HTTP-välityspalvelimet ja WWW-selaimien välimuistit.

HTTP 1.1 version suosittama malli välimuistin käytölle on seuraavanlainen. Jos millään selaimen ja HTTP-palvelimen välillä toimivalla välimuistilla on olemassa versio resurssista, jota yritetään ladata palvelimelta, on tarkistettava aina, että kyseinen versio resurssista on tuore. HTTP 1.1:n määrittely kutsuu tätä prosessia nimellä välimuistin validointi [HTTP/1.1 kappale 13.3].

Kun palvelin ensimmäisen kerran palauttaa tietyn resurssin, se liittää resurssiin HTTP-otsikoihin tarkisteen, jonka avulla välimuisti voi myöhemmin tarkistaa tiedon oikeellisuuden. Tarkiste on usein kyseisen resurssin luontipäivämäärä. Seuraavalla kerralla välimuisti ei hae automaattisesti koko resurssia palvelimelta vaan lähettääkin palvelimelle ehdollisen pyynnön. Ehdollinen pyyntö ei eroa normaalista HTTP-pyynnöstä muuten kuin sillä, että siihen on lisätty HTTP-otsikko, jossa on tieto välimuistissa olevan resurssin tarkisteesta.



Kuva 12: Välimuistin käyttötapahtuma.

Kuvassa 12 on esitetty välimuistin käyttötapahtuma, jossa kaksi käyttäjää lataavat saman resurssin samalta palvelimelta. Tässä tilanteessa sivun myöhemmin lataava käyttäjä 2 lataa sivun HTML-koodin itse asiassa HTTP-välityspalvelimelta eikä itse HTTP-palvelimelta. Jos käyttäjä 2:n toimia pyrittäisiin taltioimaan HTTP-välityspalvelimen 2 avulla, ei HTTP-vastauksen viestiosaan tallentuisi mitään. Tämän välttämiseksi on HTTP-otsikoita muutettava niin, että HTTP-välityspalvelin ilmoittaa aina välittäessään resursseja, ettei resurssia saa tallentaa välimuisteihin.

Välimuistin käyttö voidaan estää välimuistia ohjaavien HTTP-otsikoiden muutoksien avulla. Sekä selain että välimuistina toimiva välityspalvelin 1 voivat molemmat tallentaa ladatun sivun välimuistiinsa. Välimuistiin tallentaminen voidaan estää tiedonkeräystä hoitavan välityspalvelimen toimesta lisäämällä vastaukseen otsikot:

```
1: Expires: Mon, 15 May 1989 12:45:00 GMT
2: Last-Modified: Sat, 15 May 2010 12:45:00 GMT
3: Cache-control: no-cache
4: Warning: Heuristic expiration
```

Ensimmäisellä rivillä kerrotaan selaimelle, että kyseinen HTTP-resurssi vanheni vuonna 1989. Toisella rivillä kerrotaan sivun olevan muokattu tulevaisuudessa vuonna 2010. Kolmannella rivillä ilmoitetaan että kyseenomaista resurssia ei saa tallentaa välimuistiin. Neljännellä rivillä ilmoitetaan selaimelle, että HTTP-välityspalvelin on itse päättänyt välimuistiin tallennetun resurssin päiväyksen. Uudet Internet Explorer 5.x, 6.x sekä Mozilla 1.x selaimet osaavat käsitellä jokaisen näistä otsikoista.

4.1.2.3. Käyttäjästä kerättävien tietojen kerääminen

Kappaleessa 3.3 esiteltiin erilaisia tiedonkeräimiä, joilla kerättiin tietoa käyttäjän ja palvelimen välisestä tietoliikenteestä. Kappaleessa havaittiin, ettei yksikään menetelmistä yksinään tarjoa kattavaa ratkaisua tiedonkeruuongelmaan, vaan sitä varten on kehitettävä jokin menetelmät yhdistävä tapa toimia.

Välityspalvelin toimii käyttäjän ja palvelimen välillä ja sillä on mahdollisuus muuttaa ja tallentaa sekä saapuvia että lähteviä HTTP-viestejä. Koska välityspalvelin kautta kulkevat kaikki HTTP-viestit, se kykenee tuottamaan viestien sisällöstä saman määrän tietoa kuin kappaleessa 3.3.1 esitellyt loki-pohjaiset tiedonkeruumenetelmät. Kuitenkaan välityspalvelimen käytölle ei olisi perusteita, jos tiedonkeruuvoima olisi identtinen loki-pohjaisiin tiedonkeräimiin verrattuna. Etuna loki-pohjaisiin tiedonkeräimiin on se, että välityspalvelimen kautta kulkevia HTTP-viestejä voidaan myös muokata. Tällöin voidaan HTTP-viestien sisältöön lisätä tiedonkeruun kannalta tärkeitä aputiedonkeräimiä.

Välityspalvelimen tarkoituksena on kerätä käyttäjästä mahdollisimman tarkasti kaikki tieto, joka HTTP-tietoliikenteen sekä tietyillä asiakaspään komentokielillä voidaan saada selville. Jotta käyttäjien jokaisesta sivulatauksesta saataisiin mahdollisimman tarkka kuva, toteutettu välityspalvelin ei toimi yksinkertaisena läpinäkyvänä välityspalvelimena vaan muokkaa selaimen ja palvelimen välillä tapahtuvaa liikennettä, jotta jokainen sivulataus sekä jokainen yksilöllinen tieto saataisiin kerättyä.

Kerätyn tiedon siirto välityspalvelimelle

HTTP-välityspalvelimeen on toteutettu lisätoiminto, jonka avulla selaimesta kerättävä tieto saadaan tallennettua välityspalvelimen tietokantaan. Se otetaan käyttöön kahdella lisäyksellä:

1. Jokaiseen sivulataukseen joissa HTTP-palvelin lähettää otsikon `Content-type: text/html` ja sisällön havaitaan olevan HTML-koodia, Java-sovelman latauskoodin, joka välittää kerätyt tiedot välityspalvelimelle.

2. JavaScript tiedonkeräimen, joka liittää Java-sovelman latauskoodiin keräämänsä tiedot tiedonsiirtoa välityspalvelimelle varten. Lisätietojen kerääminen tapahtuu kuten kappaleessa 3.3.2 JavaScript-pohjaisilla tiedonkeräimillä. Lisäarvona on mahdollisuus kytkeä tämä kerätty tieto muilla menetelmillä kerättyihin tietoihin.

Lisätietojen välittäminen välityspalvelimelle tapahtuu sivulle ladattavan Java-sovelman toimesta, joka lähettää tiedot selaimesta ja koneesta kerätyistä tiedoista sivulta poistuttaessa välityspalvelimelle. Java-sovelmalle annetaan parametritietona JavaScriptin avulla selvitetyt tiedot käyttäjästä ja tämän selaimesta sekä sivulatauksen yksilöivä tunnistenumero, jolloin sovelman välittämät tiedot selaimesta ja käyttäjän koneesta osataan kytkeä oikeaan sivulataukseen. Tällä tavoin voidaan välittää kaikki tieto välityspalvelimelle, joka on esitelty taulukossa 12. Koska Java-sovelma lähettää tiedot vasta sivulta pois siirryttäessä, saadaan tietokantaan tieto lisäksi siitä, kauan käyttäjä on sivulla viettänyt aikaa.

Käyttäjän yksilöiminen

Käyttäjän yksilöiminen perustuu transaktioon, jossa käyttäjän koneelle on asennettava jokin tunniste, joka sivulatauksien yhteydessä välitetään palvelimelle. Kun palvelin vastaanottaa tunnisteen, voi se kytkeä sivulatauksen edellisiin sivukäynteihin.

Yksinkertaisimmillaan tämä tapahtuu käyttämällä eväste-mekanismia, joka toteuttaa edellä kuvatun transaktion täydellisesti ja on tarkoitettukin juuri pienimuotoisen tiedon tallentamiseen käyttäjän koneelle. Tästä johtuen välityspalvelimessakin käytetään käyttäjän yksilöimiseen evästeitä. Käytännössä HTTP-välityspalvelin lähettää käyttäjän koneelle kaksi erillistä evästettä, joista toinen on pitkäaikaista ja toinen istuntokohtaista tarkkailua varten, samalla tavoin kuten kappaleessa 3.3.2 on esitelty. Evästeiden lähetystä palvelimelle voidaan rajata koskemaan vain tiettyjä hakemistoja. Koska käyttäjä halutaan tunnistaa samaksi mahdollisimman laajalla alueella, molempien evästeiden poluiksi merkataan "/", jolloin kaikki kyseenomaiselta palvelimelta ladattavat resurssit käyttävät samoja evästeitä. Kun välityspalvelin määrittää evästeet, voidaan kerätty tietosisältö tiedonanalysoinnin rajapinnassa lajitella automaattisesti käyttäjäkohtaisiin tietorakenteisiin.

Evästeet asetetaan seuraavilla HTTP-otsikoilla:

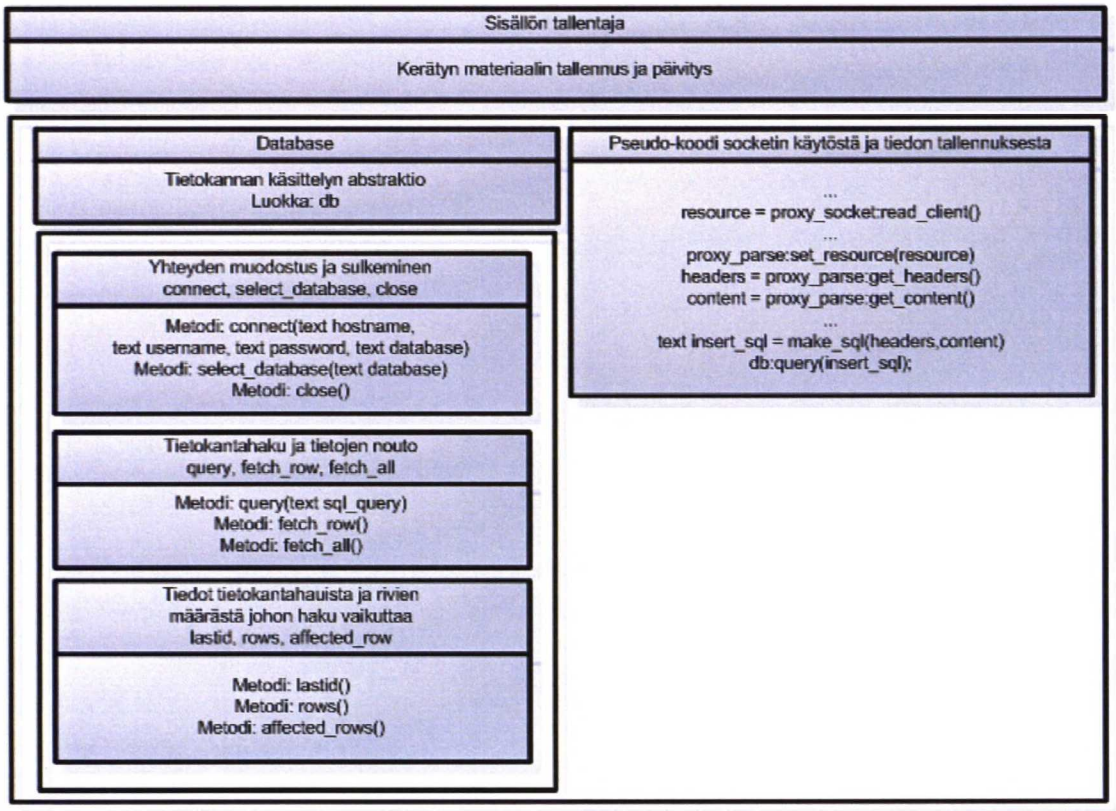
```
1: Set-Cookie:
__proxycookie=c5b7d438f60c08bf192d93f231ac7fb1; path=/
2: Set-Cookie:
__proxycookie2=26d552c013fd15936e4eb1c918c85c5a;
   expires=Thu, 14 Jan 2016 07:09:04 +0200; path=/
```

Evästeet eivät ole ainut mahdollinen menetelmä käyttäjän yksilöimiseen. Käyttäjän yksilöimiseen voidaan käyttää uusien selaimien tukemaa välimuistitoiminnallisuutta, joka esiteltiin kappaleessa 4.1.2.2. Kun selain hakee palvelimelta tietyn resurssin palauttaa palvelin hänelle samalla `Last-modified` HTTP-otsikon eli tiedon, milloin resurssia on viimeksi muokattu. Tätä tietoa voidaan käyttää käyttäjän yksilöimistunnisteena [MEANTIME]. Koska `Last-modified` tieto pohjaa monissa järjestelmissä Unixin timestamp aikaleimaan voidaan `Last-modified` tiedon avulla yksilöidä "vain" hieman yli miljardi käyttäjää (20.10.2004). Yksilöinnin rajoitus johtuu siitä, ettei `Last-modified` tiedon arvo voi sijaita tulevaisuudessa. Käyttäjä voidaan tunnistaa kun hänen selaimensa kysyy seuraavalla sivulatauksella palvelimelta onko resurssi muuttunut tunnisteena toimivan päiväyksen jälkeen.

4.1.3. Sisällön tallentaja

Sisällön tallentaja tallentaa sisällön muodostajan ja -muokkaajan rakentaman tietosisällön tietokantaan. Jotta HTTP-välityspalvelimen käyttömahdollisuudet olisivat mahdollisimman laajat, ei alkuperäisiä HTTP-otsikoita ja sivun sisältöä poisteta vaan ne tallennetaan myös tietokantaan. Näin voidaan myöhemmin tarkistaa mitä muutoksia sivun alkuperäiseen lähdekoodiin on tehty testin aikana ja tarvittaessa toistaa testitilanne sekä alkuperäisiä että muutettuja sivun sisältöä käyttäen.

Sisällön tallentajan tallentamat tiedot saadaan käyttöön omiin sovelluksiin kappaleessa 4.3 esitellyn tiedonanalysoinnin rajapinnan kautta.



Kuva 13: Sisällön tallentaja -osan kuvaus.

4.2. Tapahtumien kerääminen

HTTP-välityspalvelin tallentaa tiedot HTTP-kyselystä ja -vastauksesta tietokantaan. Tietokantatallennus on pyritty pitämään mahdollisimman yksinkertaisena ja suoraviivaisena, jotta mahdolliset järjestelmän sisäisten tietorakenteiden muutokset eivät aiheuttaisi ongelmia tietojen jälleenkäsittelyyn.

Kerätty tieto tallennetaan tietokantaan siinä muodossa kuin se WWW-selaimelta ja HTTP-palvelimelta saadaan. Muokatut HTTP-otsikot ja sivujen sisältö tallennetaan myös muodossa, jossa ne voidaan syöttää suoraan selainohjelmalle tietokannasta ilman että niitä tarvitsee käsitellä erikseen. Käytännössä tällöin tallennukset ja haut kantaan ovat hyvin nopeita operaatioita.

Kerätty tieto taltioidaan tietokantatauluun siten että HTTP-välityspalvelimeen toteutettu tiedonkerääjä tallentaa sinne taulukossa 18 esitellyt tiedot. Päivittämällä tiedonkeräintä voidaan tallentaa monipuolisempia tietorakenteita. Tässä diplomityössä ei kuitenkaan ole tavoitteena toteuttaa kaikkivoipaa tiedonkeräintä. Tarkoituksena on pohtia miten erityyppisten tiedonkeruumenetelmien avulla kerättävää materiaalia voidaan kytkeä mahdollisimman helposti välityspalvelimen tallentamaan tietoon.

Tietokantataulun kenttänimi	Kentän kuvaus	Tietotyyppi
id	Tietokantarivin yksilöivä ID-numero.	int(11)
arrival	Aika, jolloin sivulle saavuttu.	datetime
departure	Aika, jolloin sivulta siirrytty pois.	datetime
url	Ladatun sivun URL-osoite.	text
remote_addr	Välityspalvelimen näkemä selaimen IP-osoite.	varchar(255)
cookie	Sivulle lähetetyt evästeet (HTTP-otsikko `Cookie`).	text
set-cookie	Sivuston lähettämät pyynnöt lisätä uusi eväste. (HTTP-otsikko `Set-Cookie`).	text
in_headers_in	WWW-selaimen HTTP-välityspalvelimelle lähettämän HTTP-pyynnön otsikot.	text
in_headers_out	HTTP-välityspalvelimen HTTP-palvelimelle lähettämän HTTP-pyynnön otsikot. Mahdolliset muutokset tapahtuneet sisällön muokkaajan toimesta.	text
out_headers_in	HTTP-palvelimen lähettämän HTTP-vastauksen otsikot HTTP-välityspalvelimelle.	text
out_headers_out	HTTP-välityspalvelimen lähettämän HTTP-vastauksen otsikot WWW-selaimelle. Sisältää myös mahdolliset sisällön muokkaajan tekemät muutokset.	text
in_content_in	Selaimen HTTP-välityspalvelimelle lähettämän HTTP-pyynnön sisältö. GET-palvelupyynnössä kenttä on tyhjä ja POST-muotoisessa palvelupyynnössä sisältää arvot ja muuttujat HTTP-palvelimelle.	text
in_content_out	HTTP-välityspalvelimen HTTP-palvelimelle lähettämän HTTP-pyynnön sisältö. Sisältää myös mahdolliset sisällön muokkaajan tekemät muutokset.	text
out_content_in	HTTP-palvelimen lähettämän HTTP-vastauksen sisältö HTTP-välityspalvelimelle.	text
out_content_out	HTTP-välityspalvelimen lähettämän HTTP-vastauksen sisältö WWW-selaimelle. Sisältää myös mahdolliset sisällön muokkaajan tekemät muutokset.	text

Taulukko 18: Välityspalvelimen käyttämän tietokantataulun kuvaus.

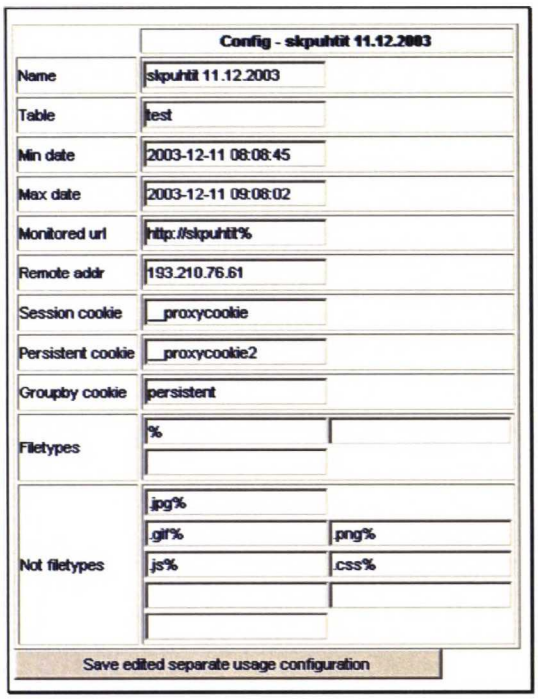
4.3. Tiedonanalysoinnin rajapinta

Kappaleessa 3.3 esiteltiin kolme erilaista menetelmää kerätä tietoa WWW-pohjaisten järjestelmien käytöstä automatisoidusti. Nämä olivat lokeihin, evästeisiin ja JavaScript:n sekä referer-viittauksiin pohjautuvat tiedonkeruumenetelmät. Jokaisen menetelmän käyttöönotto on suhteellisen yksinkertaista. Koska HTTP-välityspalvelin tarjoaa mahdollisuuden tallentaa tietoa kaikkien edellä mainittujen tiedonkeruumenetelmien avulla, on kerättävän tiedon määrä hyvin suuri. Tästä johtuen on tiedonanalysoinnin rajapinnan tarjottava tiedon esikäsittelyä ennen varsinaisia analyysejä [USABILITYENGINEERING, s.171]. HTTP-välityspalvelimeen on toteutettu rajapinta, joka tiedonkeruuvaiheessa tallennettujen yksilöivien tunnisteiden sekä IP-osoitteiden avulla jaottelee käyttäjien sivulataukset ensin käyttäjäkohtaisiin rakenteisiin, jotka sisältävät istuntokohtaiset tietorakenteet.

Tietorakenne luodaan käyttäen samoja luokkia, jotka rakentavat HTTP-välityspalvelimelle HTTP-pyynnöstä tai –vastauksesta tietorakenteen. HTTP-välityspalvelin tarvitsee monia tietoja päättäessään, miten HTTP-pyyntöjä ja -vastauksia käsitellään. Tätä varten tietojen pohjalta luodaan tietorakenne, johon päätökset voi perustaa. Kun resurssista on luotu tietorakenne, ryhmittelee tiedonanalysoinnin rajapinta tietorakenteet vielä käyttäjän yksilöivien tunnuksien perusteella.

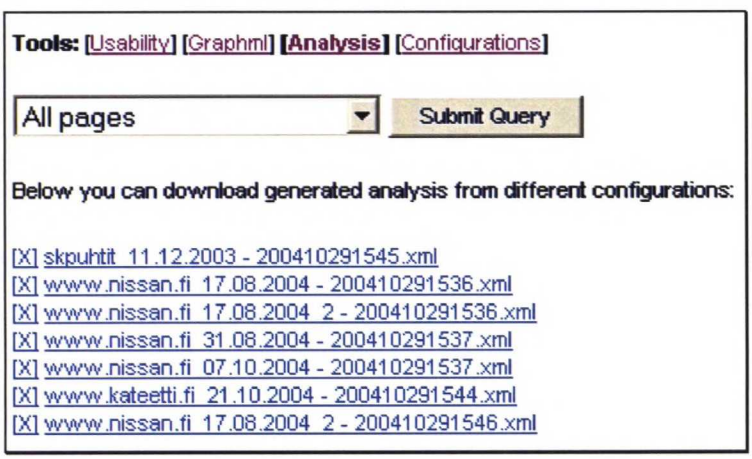
Tietorakenteen XML-tiedosto luodaan analysointirajapinnassa kahdessa vaiheessa:

1. Rajataan tietorakenteeseen haluttavat tiedot toteutetun konfiguraatioeditorin avulla, jossa määritetään mitkä tiedot tietokannassa rajaavat tietorakenteeseen pääsyä.



Kuva 14: Konfiguraatioeditori.

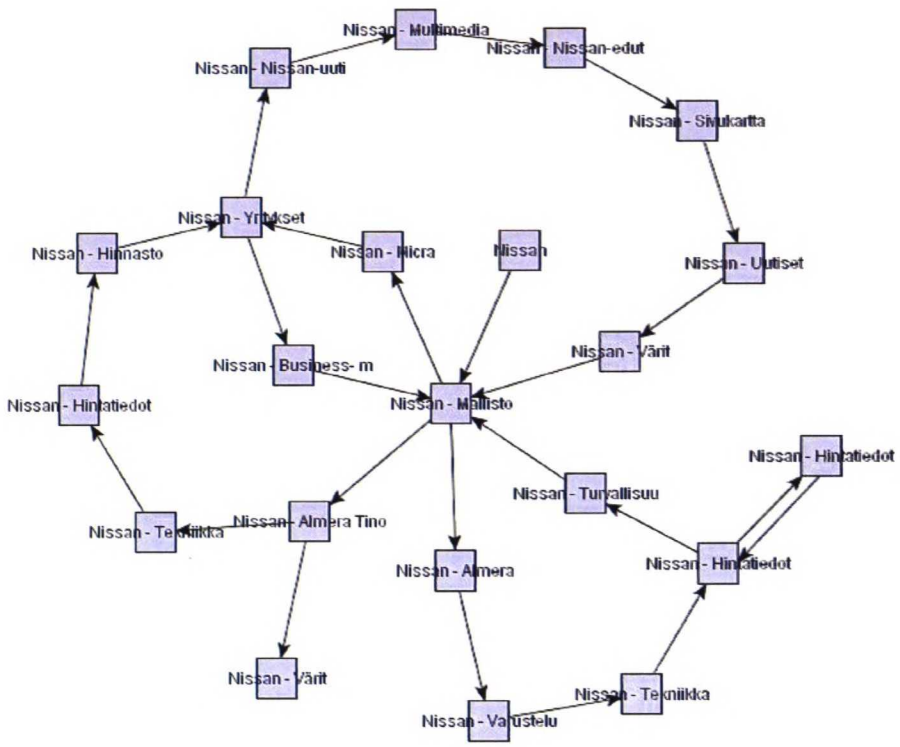
2. HTTP-välityspalvelimen analysointirajapinta luo XML-dokumentin. Tämä tapahtuu joko HTML-lomakkeen avulla tai, jos tiedon analysointi suoritetaan jossain ulkoisessa järjestelmässä, voidaan sivulle tehdä myös suoraan POST-muotoinen HTTP-pyyntö. HTTP-pyynnössä ainoana kenttänä on usedconfiguration ja sen arvona halutun konfiguraation nimi.



Kuva 15: Analyysitiedostot.

Tiedonanalysoinnin rajapinta tuottaa materiaalin omassa rakenteisessa XML-tiedostomuodossaan. Mahdolliset muut esitysmuodot, kuten kappaleessa 3.3.1.1 esitellyt loki-pohjaiset tiedostomuodot voidaan luoda rakennetun XML-tiedoston pohjalta. Muista mahdollisista formaateista joihin tietoa voidaan tietorakenteen pohjalta viedä, kannattaa mainita XML-pohjainen GraphML. Valmiita tiedostoformaatteja kannattaa hyödyntää mahdollisuuksien mukaan, koska niihin on rakennettu tiedon analysointisovelluksia, jotka ovat usein hyvin pitkälle tuotteistettuja ja niiden tarjoaman toiminnallisuuden toteuttaminen veisi pitkään.

HTTP-välityspalvelin tallentaa jokaisesta sivulatauksesta kaikki HTTP-otsikot. Tällöin tallentuu myös – jos selain on tiedon palvelimelle lähettänyt – Referer HTTP-otsikko, eli edellisen sivun URL-osoite. Tällöin kerätyn tiedon pohjalta voidaan rakentaa graafeja mahdollistamaan käyttäjän liikkeiden analysointia. GraphML tarjoaa menetelmän tiedon visualisoimiseen graafien avulla [GRAPHML]. GraphML saadaan luotua toteutetun XML-tietorakenteen pohjalta hyvin helposti. Graafien tarkastelua ja tarkempaa analysointia varten on olemassa erittäin monipuolinen Java-pohjainen yEd Graph Editor [YED].



Kuva 16: GraphML:n pohjalta yEd Graph Editorilla luotu graafi sivuston käytöstä.

4.3.1. Käytettävyystestit

Tiedonanalysoinnin rajapinnan avulla voidaan tuottaa graafeja sivuston käytöstä ja näin tutkia miten käyttäjät sivustolla liikkuvat. Näin havaitaan miltä sivuilta käyttäjät siirtyvät pois sivustolta ja mille sivustolle käyttäjät saapuvat. Näiden tietojen pohjalta voidaan mm. suunnitella sivuston rakennetta ja parantaa samalla linkitystä ja navigaatiota tukemaan käyttäjien käyttötapoja.

Välityspalvelimen tallentamat sivulataukset tukevat myöhempää käytettävyystesteistä kerättyjen materiaalien liittämistä yhteen. Kun välityspalvelin on tallentanut jokaiselle sivulataukselle myös saapumis- ja poistumisajankohdat, on kerätyn materiaalin yhdistäminen oikeisiin sivulatauksiin vaivatonta.

4.3.2. Sovellustestaus

Tiedonanalysoinnin rajapinnan kautta saadaan selville kaikki palvelimen ja selaimen välillä kulkenut tieto. Näiden tietojen avulla voidaan myös selvittää ongelmia joita syntyy WWW-pohjaisia järjestelmiä ohjelmoitaessa. Kun välityspalvelin tallentaa ja näyttää kaikki tiedot, jotka se on saanut selaimelta ja palvelimelta, voidaan selvittää missä kohdassa polkua palvelimen ja selaimen välillä on ongelma.

Ongelmien ratkaisu voi tapahtua esimerkiksi yhdistelemällä HTTP-kutsussa siirtyviä parametreja ja muodostamalla niistä kokonaisuuksia. Esimerkiksi, jos tietty muuttuja on muotoa "123321-12332" ja toinen muuttuja muotoa "123€" ja tiedetään, että kyseessä on tietty sivu sovelluksessa, voidaan olettaa, että kyseessä on pankkimaksu yms. Näin saadaan sovelluksen testauksessa tehdyt tapahtumat kartoitettua.

4.3.3. Tilastointi ja raportointi

Tiedonanalysoinnin rajapinnan kautta voidaan myös luoda erilaisia raportteja sivuston käytöstä ja mahdollisista tapahtumista. Tällä tavoin voidaan tehdä tilastointia miten eri sivuja on ladattu ja kuinka monta kertaa tiettyä resurssia on tietyillä POST- tai GET-muotoisilla parametreilla noudettu. Tieto eri resurssien latauksista on tärkeä, jos halutaan luoda esimerkiksi laskutusta varten tietoja järjestelmän käytöstä.

Esimerkkinä voidaan käyttää tekstiviestin lähetyspalvelua ulkoisen palveluntarjoajan HTTP-palvelimelta. Palvelimelle saapuvat kaikki tietyn yrityksen sovelluksen käyttäjät samasta IP-osoitteesta palomuurin takaa. Tällaisessa tilanteessa palveluntarjoaja ei voi tehdä erittelevää laskua asiakkaalle eri käyttäjien tekemistä tekstiviestien lähetyksistä. Kun kaikki käyttäjät käyttävät välityspalvelinta, joka sijaitsee samassa verkossa käyttäjien kanssa, saadaan tallennettua jokainen käyttäjä sekä heidän järjestelmän avulla lähettämien tekstiviestien lukumäärä.

5. PROJEKTIHALLINTAJÄRJESTELMÄN KÄYTETTÄVYYSARVIOINTI

Yhtenä osana tätä diplomityötä oli testata HTTP-välityspalvelimen käyttöä käytettävyysarvioinnin osana. Testikohteena oli Salomaa-yhtiöiden käyttämä projektinhallintajärjestelmä. HTTP-välityspalvelinta käytetään osana käytettävyystestausta tallentamaan käytettävyystestissä syntyvää materiaalia myöhemmin toteutettavassa käytettävyystestien tuloksien analysoinnissa.

Toteutetussa analysointisovelluksessa yhdistetään heuristisen arvioinnin pohjalta saadut ongelmakohdat kerättyyn videomateriaaliin, muistiinpanoihin sekä välityspalvelimen käytöstä keräämään dataan. Heuristisen arvioinnin menetelmistä käytettiin tunnettua Nielsenin 10 heuristiikkaa [10HEURISTICS].

Kokemukset ja havainnot käytettävyystesteistä sekä välityspalvelimesta käytöstä on esitelty kappaleessa 6.1.

5.1. Testitilanne

Testauksen kohteena on vuoden käytössä ollut projektinhallintajärjestelmä, jossa voidaan suorittaa eri toimintoja liittyen laskutukseen, raportointiin sekä tuntikirjanpitoon. Järjestelmä kehitettiin varsin pitkälti räätälöitynä järjestelmänä Salomaa-konsernin käyttöön.

Ongelmaksi muodostui käytettävyystestauksen puuttuminen järjestelmän toteutusvaiheessa. Tämän vuoksi ei nyt tehtävää käytettävyystestausta varten ole olemassa pohjamateriaalia järjestelmästä, joka tukisi testien tekemistä. Järjestelmän ylläpidosta vastaa Salomaa-konsernin yhteistyökumppani, joka on myös implementoinut projektinhallintajärjestelmän. Vuoden aikana ilmenneet ongelmat on pyritty niputtamaan isommiksi päivitysryppäiksi, mutta pienimuotoiset päivityksetkin ovat yleisiä. Näin on saatu suurimmat lastentaudit karsittua. Käytettävyystesteillä halutaan saada selvyys siihen, piileekö järjestelmän käyttöliittymässä ja käytössä vielä ongelmia, joita ei ole havaittu normaalissa käytössä käyttäjien toimesta.

Testitilanteissa on paikalla kaksi henkilöä: testin ohjaava henkilö (ohjaaja) sekä testin suorittava henkilö (käyttäjä). Testin ohjaava henkilö asettaa selaimeen HTTP-välityspalvelimen käyttöön. Testitilanteessa testin ohjaaja tallentaa testitilanteen tapahtumia käyttämällä videokameraa sekä muistiinpanovälineitä.

5.2. Testitehtävät

Projektinhallintajärjestelmän käytöstä testattavat tehtävät/asiat jakautuivat kolmeen pääkategoriaan:

1. Tiedon lisääminen ja korjaaminen:

   Käyttäjää pyydettiin lisäämään järjestelmään projektiin liittyviä tietoja sekä muuttamaan niitä. Mahdollisiksi ongelmakohdiksi arvioitiin etukäteen mm. vanhojen/vieraiden projektien löytäminen sekä sen päättelyä mistä tiettyä tietoa voidaan ylläpitää.

2. Tiedon etsintä:

   Käyttäjää pyydettiin etsimään tiettyjä järjestelmään etukäteen tallennettuja tietoja projekteista. Mahdollisiksi ongelmakohdiksi arvioitiin etukäteen mm. harvemmin tarvittavien tietojen löytyminen sekä ohjeiden käytön vähäisyys.

3. Järjestelmän muistettavuus:

   Testin jälkeen pyydettiin käyttäjää toistamaan osa tehtävistä ääneen. Tällä pyrittiin selvittämään kuinka hyvin käyttäjä muistaa järjestelmässä tehtyjä harvinaisempia tehtäviä testitilanteen jälkeen.

Eri testitehtävät vaativat varsin erityyppistä analysointia HTTP-välityspalvelimen keräämästä datasta. Tiedon lisääminen ja korjaaminen on hyvin suoraviivainen operaatio ja sen onnistumista voidaan tutkia esimerkiksi laskemalla aikaa tehtävän aloituksesta siihen kun HTTP-palvelimelle lähetetään tietokentät oikeassa muodossa. Tiedon etsintä on varsin vaativa operaatio analysoinnin kannalta, järjestelmä ei kykene havaitsemaan onko käyttäjä varsinaisesti löytänyt tietoa vaikka hän olisikin oikealla sivulla käynyt. Tätä varten on käytettävä kirjoitettuja muistiinpanoja sekä tallennettua videomateriaalia. Järjestelmän muistettavuus tullaan taltioimaan järjestelmään myöskin vasta tiedon analysointivaiheessa samalla kun videomateriaalia indeksoidaan.

5.3. Välityspalvelimen käyttö ja kerättävän tiedon konfigurointi

HTTP-välityspalvelinta ja toteutettua analysointisovellusta käytetään mahdollistamaan yhden ihmisen toteuttama käytettävyystestaus projektinhallintajärjestelmästä. Kun yksi ihminen on vastuussa sekä videomateriaalin tuottamisesta, testin ohjaamisesta että muistiinpanojen tekemisestä on tärkeää että testitilannetta taltioidaan myös muuten kuin testin pitäjän toimesta.

Ennen käytettävyystestien alkua HTTP-välityspalvelimelle konfiguroitiin seuraavat tehtävät tiedon keräämistä ja analysointia varten. Ennen testitilanteita HTTP-välityspalvelimen tallentamat tiedot testitilanteesta päätettiin käymällä jokainen testitehtävä erikseen läpi ja pohtimalla mitä tietoja projektinhallintajärjestelmän käytöstä tarvitaan, jotta testitilanne voidaan myöhemmin mahdollisimman tarkkaan toistaa. Testin toistaminen tapahtuu käytännössä yhteydessä videomateriaalin kanssa kun videomateriaalin sisältämä tieto indeksoidaan. Nauhojen indeksointi on aikaa vievä prosessi ja sen tuottaman datan tulisi olla mahdollisimman helposti käytettävissä muita analysointeja varten.

Analysointia varten päätettiin kerätä seuraavat tiedot sekä lisätä tiettyjä HTTP-otsikoita, jotta voidaan varmistua kerätyn tiedon laadusta.

Kerättävä tieto sekä muutokset HTTP-vastaukseen	**Miten tieto kerätään**

Miten käyttäjä liikkuu sovelluksessa?	Käyttäjän liikkeet tallennetaan HTTP-välityspalvelimen toimesta tallentamalla selaimen HTTP-pyynnössä olevat POST, GET ja HEAD -muotoa olevat pyynnöt.
	
	`GET http://www.esim.fi/ HTTP/1.1`
	
	Näiden tietojen lisäksi tallennetaan HTTP-otsikko `Referer`, joka sisältää tiedon edellisestä sivusta. Näin voidaan käyttäjän vierailu myöhemmin kytkeä yhtenäiseksi tapahtumasarjaksi.
	
	`Referer: http://www.esim.fi/edellinen.html`

Kuinka kauan yhdellä sivulla on vierailtu?	Sivulla käytetty aika auttaa mm. videomateriaalin indeksointia sekä sen pohjalta voidaan tutkia kuinka kauan käyttäjä vietti aikaa per sivu etsiessään tietoa järjestelmästä. Tämän tiedon käyttö tapahtui tallentamalla sivun latauksen ajankohta sekunnin tarkkuudella.
Käyttäjien yksilöinti kerätyssä datassa.	Koska käyttäjätesti toteutettiin useammalle käyttäjälle, oli heidät kyettävä yksilöimään kerätystä materiaalista. Käyttäjien koneille asennettiin HTTP-välityspalvelimen asettama eväste, joka yksilöi käyttäjän järjestelmälle, lisäämällä HTTP-palvelimen antamaan HTTP-vastaukseen otsikot: `Set-Cookie: __user=YKSILÖIVÄ_TUNNISTE; expires=Thu, 14 Jan 2016 07:09:04 +0200; path=/` Tämän lisäksi tallennetaan käyttäjän IP-osoite, tilannetta varten jossa käyttäjän kone syystä tai toisesta hylkää evästeen.

Välimuistien käytön ehkäisy.	Jokaisen sivulatauksen on tapahduttava aina HTTP-palvelimelta eikä esimerkiksi selaimen välimuistista. Jos sivulataukset eivät tallennu ei videomateriaalin kytkentää voida suorittaa riittävällä tarkkuudella ja käyttäjän liikkeiden analysointi järjestelmässä estyy. Jotta jokainen sivu ladattaisiin suoraan palvelimelta, käyttämättä selaimen välimuistia lisätään HTTP-vastaukseen otsikot:
	
	`Cache-control: no-cache`
	`Expires: Mon, 15 May 1989 12:45:00 GMT`
	`Last-Modified: Sat, 15 May 2010 12:45:00 GMT`
	`Warning: 113 Heuristic expiration`

Taulukko 19: Tallennettavat tiedot sekä muutokset HTTP-vastaukseen.

5.4. Testitilanteiden analysointi

Testitilanteista tallennettiin kolmea eri materiaalia myöhempää analysointia varten: muistiinpanoja, videomateriaalia sekä HTTP-välityspalvelimen tallentama testitapahtuma. Jotta testitilanteiden analysoiminen olisi mahdollista, on nämä kolmessa eri formaatissa olevat tiedot saatava yhdistettyä myöhempää prosessointia varten.

Tiedon myöhemmän prosessoinnin helpottamista varten päätettiin kerätyt tiedot kytkeä välityspalvelimen tallentamiin sivulatauksiin. Tarvittavat analysointitehtävät ovat tästä johdettuna aikajärjestyksessä:

1. Muistiinpanojen läpikäyminen ja niiden yhdistäminen sivulatauksiin.
2. Videomateriaalin indeksointi ja indeksoitujen tapahtumien kytkentä sivulatauksiin.
3. Sivulatauksiin kytkettyjen tapahtumien kytkeminen Nielsenin 10 heuristiikkaan.
4. Kun tapahtumat on kytketty Nielsenin heuristiikkoihin, luodaan tarvittaessa havaittuihin ongelmakohtiin tiedonanalysoinnin rajapinnan avulla tarkempi analysointisovellus selvittämään kyseenomaista ongelmaa tai käytetään toteutettua tiedon analysointisovellusta.

5.5. Tiedon analysointisovellus

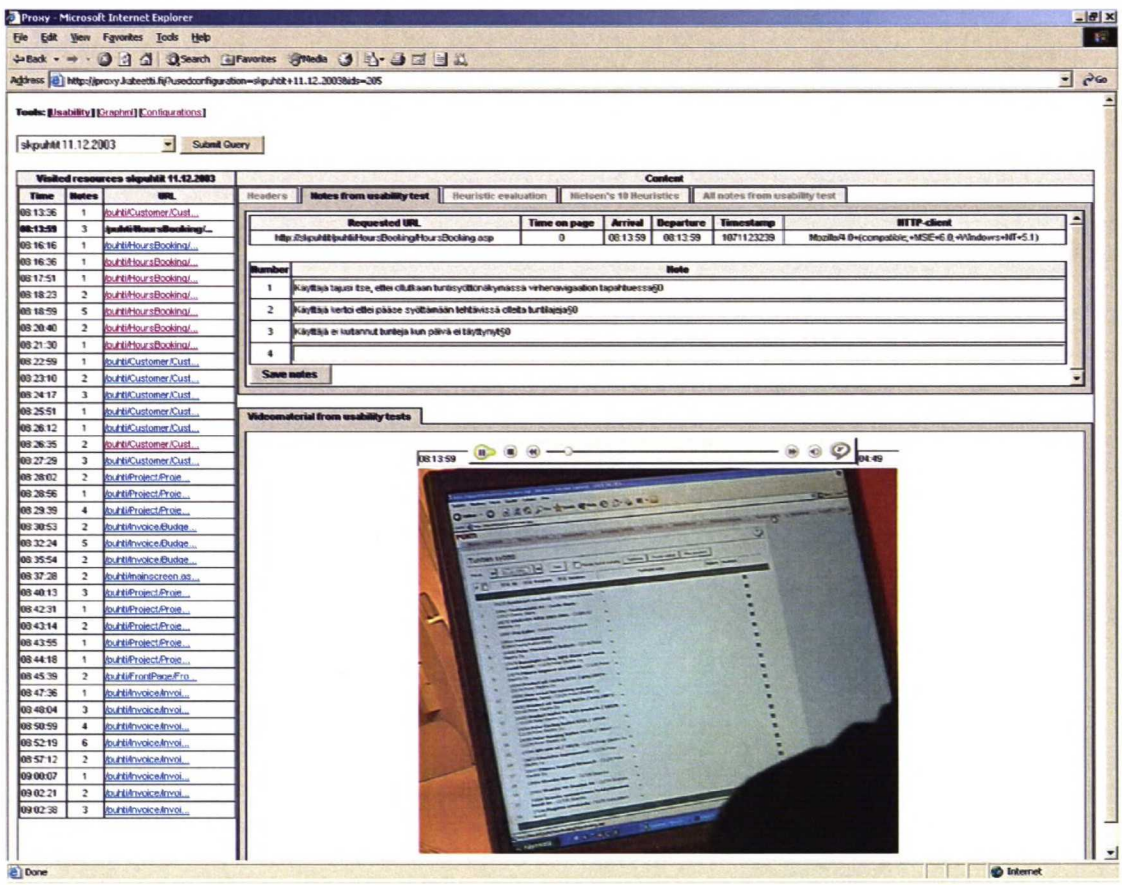
Toteutettu tiedon analysointisovellus tarjoaa mahdollisuuden kytkeä testitilanteessa tallennettuja tietoja sivulataukseen ja luokitella niitä Nielsenin heuristiikkojen pohjalta.

Tiedon analysointisovelluksen käyttö tiedon analysointiin vaatii muutamia välivaiheita edellä mainittujen analysointitehtävien lisäksi:

1. HTTP-välityspalvelimelle tallennetusta tiedosta pitää erotella tietyn käyttäjän sivulataukset ja näihin liitetyt tiedot.

2. Tuotettu videomateriaali on siirrettävä tietokoneelle sekä muutettava RealNetwork:n toteuttamaan Real Media Variable Bitrate (RMVB) tiedostomuotoon, jotta video voidaan liittää tiedon analysointisovellukseen.

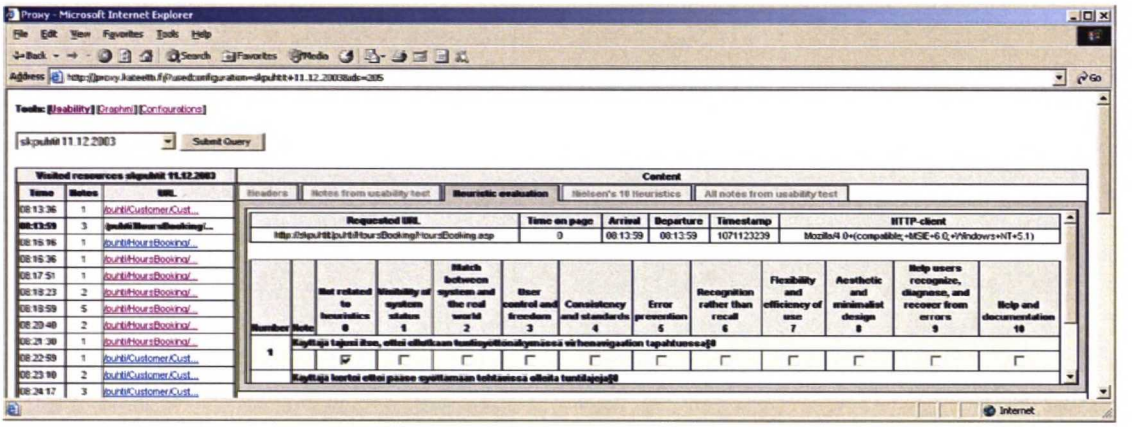
3. Videon alkamishetki on kytkettävä HTTP-välityspalvelimen keräämään tietoon, jotta videon kelaaminen sivulatauksia klikkaamalla toimisi.



Kuva 17: Tapahtumien kirjaaminen tiedon analysointisovellukseen.

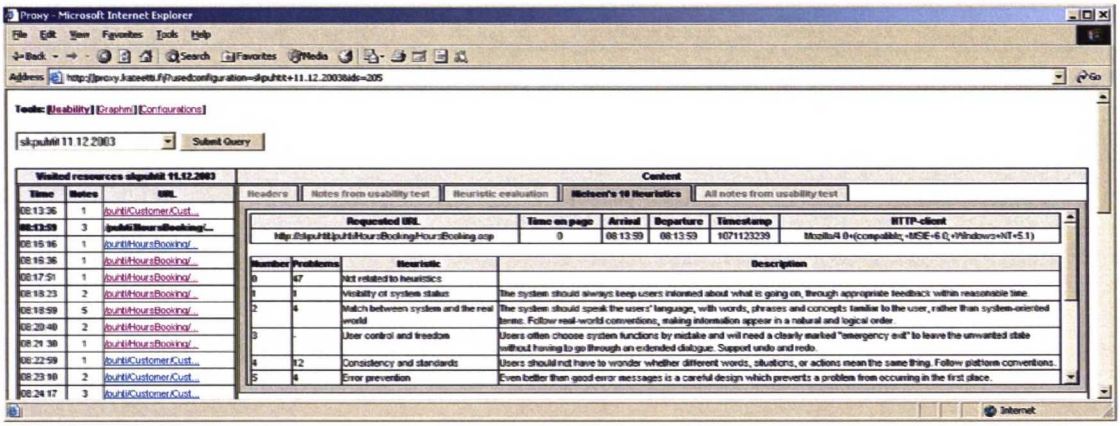
Edellä mainittujen vaiheiden jälkeen voidaan aloittaa varsinainen tietojen indeksointi ja tapahtumien tallentaminen. Tietojen indeksointi ja tapahtumien tallentaminen tapahtuu samalla tavalla. Ensin sivun yläreunasta valitaan oikea käyttötapahtuma. Tämän jälkeen alkaa tapahtumasta kerätty videomateriaali pyöriä selaimessa ja testin ohjaaja voi alkaa syöttämään sekä nauhalla että muistiinpanoissaan olevia tapahtumia analysointisovellukselle. Klikkaamalla vasemmalla olevia linkkejä pääsee testin ohjaaja seuraamaan mitä kyseenomaisella sivulatauksella testissä tapahtui ja mitä tietoja käyttäjästä on järjestelmään tallennettu, tämän lisäksi hän näkee tietysti itsensä kirjoittamat tapahtumat. Sivun latauksesta kerättävät tiedot on esitelty taulukossa 13.

Testin ohjaaja käytyä kaiken testitilanteessa kerätyn materiaalin lävitse ja syötettyä ne tiedon analysointisovellukseen alkaa varsinainen analysointivaihe. Tässä vaiheessa sovellukseen kirjatut tapahtumat kytketään Nielsenin heuristiikkoihin. Tämä tapahtuu yksinkertaisesti valitsemalla sivu, jonka tiedot halutaan kytkeä heuristiikkoihin ja valitaan ne heuristiikat joihin tapahtuma liittyy.



Kuva 18: Tapahtumien kytkeminen heuristiikkoihin analysointisovelluksessa.

Kun kaikki tapahtumat on kirjattu sovellukseen sekä ne on kytketty Nielsenin heuristiikkoihin voi testin ohjaaja käydä tarkistamassa analysointisovelluksen tekemän yhteenvedon siitä, mistä eri heuristiikoista järjestelmässä on poikettu.



Kuva 19: Yhteenveto tiedon analysointisovelluksessa heuristiikoista.

6. JOHTOPÄÄTÖKSET

Tässä kappaleessa kuvataan toteutetun HTTP-välityspalvelimen käytön sekä myöhemmän tiedon analysoinnin pohjalta havaittuja johtopäätöksiä. Esitellään kokemuksia järjestelmän käytöstä, havaittuja ongelmia ja parannusehdotuksia.

Varsinaisen järjestelmän parannusehdotusten lisäksi esitellään kehityskohteita, jotta tiedonanalysoinnin rajapintaa ja sen keräämää tietoa voitaisiin käyttää mahdollisimman helposti myös muun kaltaisissa projekteissa kuin kappaleessa 5 esitellyssä käytettävyysarvioinnissa.

6.1. Kokemukset järjestelmän käytöstä

Toteutettu välityspalvelin ja tiedonanalysoinnin rajapinta tarjosivat hyvät edellytykset käyttäjän tekemien tietojen tallentamiseen tietokantaan sekä niiden myöhempää analysointia varten. Lähinnä ongelmia aiheuttivat mahdolliset jatkoselvitykset, joita ei ennen käytettävyystestejä ollut osattu huomioida.

Seuraavissa kappaleissa esitellään testeissä tehtyjä havaintoja, ongelmia sekä niiden pohjalta tehtyjä muutoksia välityspalvelimeen.

6.1.1. Kerätyn tiedon rajaaminen

Käytettävyystestit saatiin suoritettua sovitussa aikataulussa ja testeistä saatiin kerättyä se data mitä testejä suunniteltaessa päätettiin kerätä. Ongelmaksi muodostui jatkotutkimuksien kannalta kerätyn tiedon rajaus. Havaittiin, että testitilanteista kerättävää tietoa ei saisi rajata. Kun tietoa rajataan, rajataan myös sen jatkokäyttöä. Jos jokaisesta testitilanteesta on kerätty sama kattava tietomäärä, voidaan kuukausia testien jälkeen analysoida kerättyä tietoa uudelleen, vaikka tutkimuksen suunnitteluhetkellä tietty kerätty tieto ei olisikaan tuntunut tarpeelliselta. Kerätyn tiedon rajaaminen hankaloittaa huomattavasti myös tiedonanalysoinnin rajapinnan käyttöä. Analysointisovelluksia toteutettaessa kun ei tässä tilanteessa voida olettaa, että joka kerta on kerätty samat tiedot.

Testeissä HTTP-välityspalvelinta käytettiin kerrallaan vain muutamalta koneelta ja tästä johtuen kerättyä tietoa syntyi vain megatavu-luokkaa. Nykyisillä relaatiotietokannoilla ja tietokoneilla tämän kokoluokan tietokannat eivät ole mikään ongelma. Vasta kun välityspalvelinta käytetään keräämään dataa sadoilta koneilta, kannattaa aloittaa kerätyn tiedon rajaaminen. Sitä ennen kannattaa kaikki mahdollinen tiedon kerääjän löytämä tieto tallentaa.

6.1.2. Sivulla vietetyn ajan selvittäminen

Testissä havaittiin että sivulla vietetyn ajan selvittäminen on hankalaa nykyisellä sivulatauksen ajankohdan ja edelliseen sivuun perustuvalla tiedon selvitysmenetelmällä. Tähän aiheuttavat ongelmia:

A. Dynaamisten sivujen luonti. Sivujen luonti voi pahimmillaan viedä jopa kymmeniä sekunteja. Selaimet alkavat esittämään tietoa käyttäjän selaimessa heti saatuaan vastaanotettua edes pienen osan sivun HTML-lähdekoodista, tästä johtuen on mahdotonta tietää tarkkaan, koska ladatun sivun varsinainen "käyttäminen" on alkanut.

B. Selainkehyksien käyttö. HTML-dokumentit toimivat selainkehyksiä käytettäessä hieman eri rooleissa. Rooleja on mm. selainkehyksien asettelu, navigaatioon liittyvät kehykset sekä itse tietosisältöön liittyvät kehykset. Välityspalvelimen kannalta on mahdotonta tietää mihin kehykseen on tallennettu varsinainen tietosisältö ja tätä kautta kytkeä näitä sivuja ketjuksi.

Sivulla vietetyn ajan selvittämiseksi toteutettiin erillinen Java-sovelma, jonka latauskoodi sisällytetään jokaiseen HTML-sivuun tiedon muokkaajan toimesta. Java-sovelma välittää tarvittavat tiedot sivulatauksesta välityspalvelimelle sivulta poistuttaessa.

6.1.3. Tiedon analysointi

Käytettävyystesteistä kerätyt tiedot otettiin käyttöön toteutettuun tiedon analysointisovellukseen, jossa varsinainen tiedon käsittely tapahtui. Tiedonanalysoinnin rajapinnan suorittama tietorakenteiden luonti ja ryhmittely auttoi huomattavasti toteutettaessa käytettävyystestien analysointisovellusta. Tarpeellisten tietojen esilletuonti sekä tietojen tutkiminen oli suoraviivaista johtuen tietorakenteesta, jossa jokainen käytöstä taltioitu tieto oli omana tietueenaan.

Sovelluksen toistama videomateriaali analysoinnin aikana helpotti muistamaan miten varsinainen testi meni. Varsinkin toisessa kahdesta testistä datan analysointi tapahtui vasta viikon päästä varsinaisesta testitilanteesta. Tiedon analysointisovelluksen avulla saatiin toteutettua siis tiedon indeksointi sekä kerätyn tiedon ryhmittely. Myöhempi tiedon analysointi tapahtui analysoimalla yksittäisiä tallennettuja tapahtumia sekä tutkimalla eri heuristiikkoihin keräytyneitä ongelmia kokonaisuuksina.

Havaittiin, että selitteiden kytkeminen sivulatauksiin olisi hyvä yleistää. Tämä tulisi tapahtua niin, että jokaisella sivulla voisi tehdä muistiinpanoja joihin voitaisiin kytkeä erilaisia selitteitä – tapahtumia. Tapahtumat voisivat olla esimerkiksi tekstiviestin lähetys HTML-lomakkeelta tai ristiriita jonkin Nielsenin heuristiikan kanssa. Jotkin tapahtumat voitaisiin erikseen ohjelmoida järjestelmän automaattista tunnistamista varten. Näitä voisivat olla mm. tiettyjen GET- ja POST-parametrien yhdistelmät, jotka tarkoittavat järjestelmässä esimerkiksi juuri edellä mainittujen tekstiviestin lähetystä. Kuitenkin Nielsenin heuristiikkojen tunnistaminen lokitiedon pohjalta on lähes mahdotonta, lähinnä voidaan havaita onko jotain toimintoa käytetty.

Havaittiin myös, että jotkin järjestelmässä tehdyt tapahtumat koostuivat useammasta eri sivusta. Jotta tietty tapahtuma järjestelmässä saatiin toteutettua, täytyi käyttäjän käydä ensin tietyllä sivulla tekemässä tietty toiminto ja siirtyä seuraavalle ja tehdä toinen toiminto. Näiden automaattinen tunnistaminen verrattuna yhdelle sivulle tapahtuvaan tiedon välitykseen on monimutkaisempaa ja vaikeammin toteutettavissa tiedonanalysoinnin rajapintaan käyttäjän konfiguroitavaksi.

6.1.4. Välityspalvelimen käyttöönotto

HTTP-välityspalvelin asentui testiin osallistuneiden käyttäjien selaimiin vaivattomasti ja käyttäjät eivät havainneet testin aikana käyttävänsä projektinhallintajärjestelmää välityspalvelimen kautta. Jos käyttäjät olisivat havainneet mahdollista hidastumista järjestelmän käytössä, olisi testien lopputuloksien analysointi ollut hankalampaa. Näin ei kuitenkaan tapahtunut.

Jos käyttäjiä olisi ollut useampi ja testitilanteesta ei olisi käyttäjille etukäteen kertoa, olisi välityspalvelimen käyttöönotto ollut hankalampaa. Tätä varten tulisi välityspalvelimen lähdekoodia muokata niin, että se emuloisi tietyissä tilanteissa paremmin itsenäistä HTTP-palvelinta. Eli välityspalvelin toimisi HTTP/1.1 määrityksen [HTTP/1.1] mukaisena läpinäkyvänä välityspalvelimena. Tällöin voitaisiin välityspalvelin ottaa tilapäisesti käyttöön vain tekemällä käyttäjien käyttämiin nimipalveluihin tilapäinen muutos ja jättämällä välityspalvelimen käyttämä nimipalvelu alkuperäiseksi. Näin HTTP-kyselyt osoittaisivat välityspalvelimelle, joka osaisi hakea oikean resurssin oikealta palvelimelta. Tämä menetelmä on esitelty tarkemmin kappaleessa 6.3.3.

6.1.5. Muut huomiot

Vaikka käytettävyystutkimus saatiinkin suoritettua yksin, on huomioitava, että käytettävyystutkimus on ryhmätyötä ja käytettävyystestit vaativat useamman kuin yhden henkilön käyttöä. Kun testeissä oli läsnä ainoastaan testiin osallistuva käyttäjä sekä testin ohjaaja, ei testitilanteissa mahdollisesti videomateriaalin sekä välityspalvelimen tallentamien tietojen ulkopuolella tapahtuneista tapahtumista voinut testin jälkeen keskustella kenenkään kanssa. Muistiinpanojen tekeminen olisi helpompaa ja käyttäjän eleet sekä muut tapahtumat huomattaisiin tarkemmin, jos testissä olisi mukana useampi testin havainnointiin ja toteuttamiseen osallistuva henkilö.

6.2. Saatiinko oikea data kerättyä?

Ennen käytettävyystestiä määritettiin mitä tietoja tulisi HTTP-välityspalvelimen testistä kerätä myöhempää tiedon analysointia varten. Nämä tiedothan olivat selattavan sivun osoite, sivulle annetut GET-muotoiset parametrit, kauan yhdellä sivulla viivytään ja käyttäjän yksilöivä tunniste. Tarkoituksena oli siis tukea videotallenteen analysointia.

Jälkikäteen havaittiin, että käytettävyystestitilanteesta olisi pitänyt kerätä myös muutamia muitakin tietoja. Tämä havaittiin, kun etsittiin ratkaisuja muihin kuin etukäteen määriteltyihin tilanteisiin. Esimerkkinä tällaisesta tilanteesta on esimerkiksi se, että varsinainen testitilanne haluttaisiin toistaa WWW-selaimessa käyttämällä tallennettuja HTML-lähdekoodeja ja kuvia. Tällöin myöhemmät järjestelmään tehtävät muutokset ja sivujen osoitteiden muutokset eivät vaikuttaisi siihen, että kyseenomaista sivua ei enää saataisi rekonstruoitua. Mahdollinen tilanne voisi olla mobiiliportaalin testaus, jossa testataan kuvien laadun vaikutusta järjestelmän käyttöön. Jos tässä tilanteessa HTML ja kuvien data olisi tallennettu, voitaisiin myöhemmin datasta rekonstruoida sivut huonommilla kuvalaaduilla ja testata sen vaikutusta sivun käyttöön. Havaittiin myös, että sivulla vietetty aika tulee laskea siitä hetkestä, kun selain on saanut sivun prosessoitavakseen, eikä siitä, kun HTTP-välityspalvelin on HTTP-pyynnön vastaanottanut. Näiden välillä voi olla hitailla yhteyksillä sekä raskaita dokumentteja ladattaessa merkittäväkin ero.

Myöhempien mahdollisesti hämärän peitossa olevien tutkimuskohteiden takia ei kerättävää tietoa kannata lähteä karsimaan. Jos testattavana on vain muutamia käyttäjiä, kykenevät nykyiset relaatiotietokannat helposti käsittelemään syntyvää tietomäärää niin, että myös mahdollisesti syntyvät jatkokehitys ja -tutkimuskohteet saadaan tutkittua.

6.3. Kehityskohteita

Tässä kappaleessa on esitetty mahdollisia välityspalvelimen tiedonanalysoinnin rajapinnan jatkokehityskohteita sekä ideoita miten toteutettua välityspalvelinta ja sen toimintaa voitaisiin kehittää.

6.3.1. Tapahtumien koostaminen useammasta sivupyynnöstä

HTTP-välityspalvelinta käytettiin testaamaan käyttäjän toimia projektinhallintajärjestelmässä erillisten tehtävien kautta. Kuitenkin parasta tietoa järjestelmän käytöstä saataisiin, kun käyttäjä ei tietäisi osallistuvansa varsinaisesti testiin, vaan hänen sovelluksen käyttöä seurattaisiin pitkällä aikavälillä. Tällöin voitaisiin havaita käyttäjän liikkeistä järjestelmässä tärkeiden sovelluksen osa-alueiden välisiä navigaatio-ongelmia.

Jotta käyttäjän toimia voitaisiin tarkasti analysoida, pitäisi toteuttaa tiedonanalysoinnin rajapinnan avulla analysointisovellus, jolla tutkittaisiin monen mahdollisesti peräkkäisen sivulatauksen pohjalta syntyviä sivuketjuja. Yksittäinen tapahtuma järjestelmässä kun voi koostua usealla eri sivulatauksella tapahtuvista HTTP-pyynnöistä.

Esimerkkinä tällaisesta operaatiosta voidaan pitää esimerkiksi tuntikirjanpitoon tuntien syöttämistä, jossa ensin valitaan tietyltä sivulta projekti ja seuraavalle sivulle täytetään kyseenomaiselle projektille tuntimäärät. Tällöin yksittäistä sivua tutkimalla ei välttämättä saada selville mihin projektiin tunteja ollaan syöttämässä.

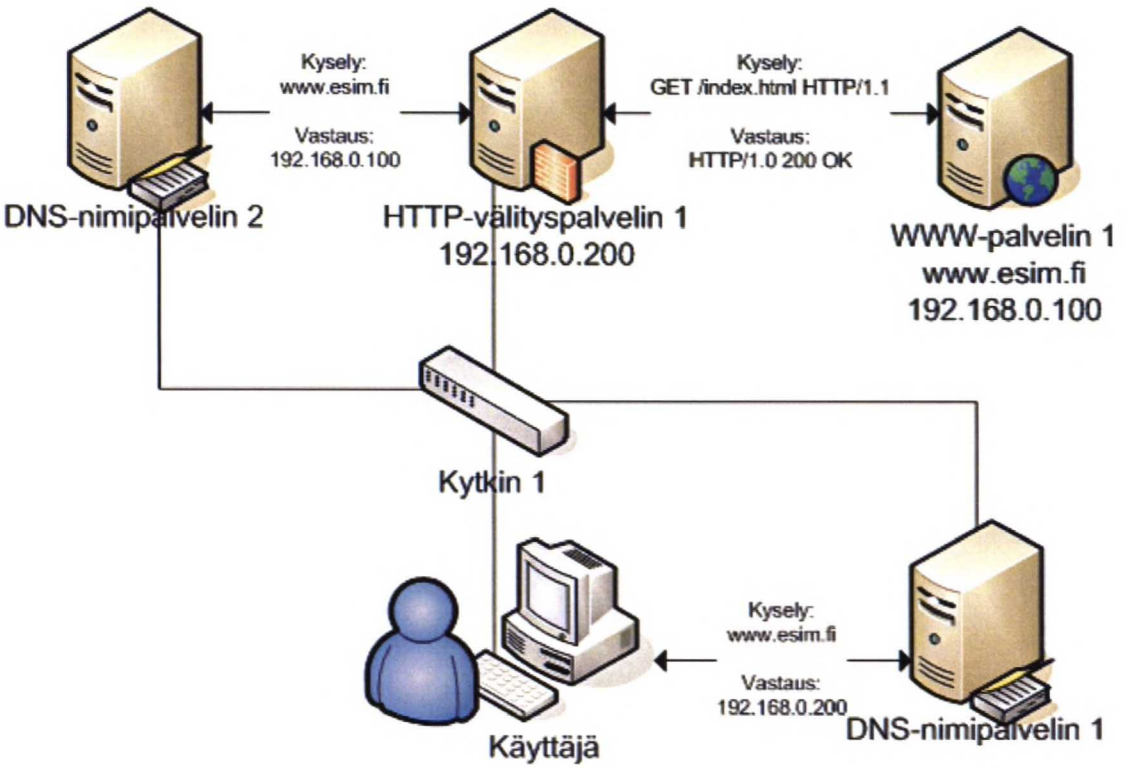
6.3.2. HTTP-viestien muokkaaminen ulkoisissa sovelluksissa

HTTP-viestien muokkaamista varten on toteutettu rajapinta, jonka implementoimalla voidaan järjestelmään liittää ulkoisia sovelluksia. Tällä hetkellä ulkoisen sovelluksen liittäminen vaatii kuitenkin, että ulkoinen sovellus on toteutettu samalla ohjelmointikielellä kuin varsinainen sovellus.

Tiedon muokkaaja -rajapintaan voisi rakentaa esimerkiksi monikäyttöisemmän SOAP-rajapinnan (Simple Object Access Protocol), jonka avulla mikä tahansa verkkopohjainen järjestelmä voi muuttaa sekä selaimen lähettämiä HTTP-pyyntöjä että palvelimen palauttamia HTTP-vastauksia.

Viestin muokkaaminen voisi hyvinkin tukea esimerkiksi PDA-laitteiden ja matkapuhelinten WWW-sivujen käyttöä. Tällöin tiedon muokkaaja muokkaisi automaattisesti sivulla esitetyt kuvat pienemmälle tarkkuudelle ja muuttaisi mahdolliset harvinaisemmat kuvatiedostomuodot selaimen ymmärtämään muotoon.

6.3.3. HTTP-välityspalvelin HTTP-palvelimeksi



Kuva 20: HTTP-palvelimen emulointi.

HTTP-välityspalvelimen käytössä on lähinnä yksi suurempi ongelma. HTTP-välityspalvelin on asetettava käyttöön jokaiseen selaimeen erikseen. Välityspalvelimen käyttöönoton useillekin koneille pitäisi laajempia testejä varten olla vaivatonta ja mahdollista ilman, että käyttäjät sitä havaitsisivat.

Välityspalvelimen käyttöönotto usealle koneelle tapahtuu nimipalvelinten asetuksia muokkaamalla. Varsinaisesti HTTP-välityspalvelimen lähdekoodiin ei tarvitse tätä varten tehdä muutoksia, koska HTTP/1.1 määrityksen mukaan välityspalvelimen on implementoitava sekä HTTP-palvelimen että WWW-selaimen toiminnallisuus. Muuttamalla selaimien käyttämän nimipalvelimen kaikkien HTTP-palvelinten osoitteet osoittamaan välityspalvelimeen, saadaan kaikki selaimet ohjaamaan HTTP-pyyntönsä välityspalvelimelle. Lähiverkossa on välityspalvelinta varten toinen nimipalvelin, joka ilmoittaa HTTP-palvelimien oikeat IP-osoitteet. Näin välityspalvelin osaa suorittaa HTTP-pyynnöt oikealla HTTP-palvelimella. WWW-selain lähettää samanlaiset kyselyt aina sekä HTTP-palvelimelle että välityspalvelimelle, joten tästä ei tarvitse huolehtia.

7. LÄHTEET

[10HEURISTICS]
Jakob Nielsen and Robert L. Mack 1994
Usability Inspection Methods, Heuristic Evaluation
Nielsen Norman Group
New York: John Wiley & Sons

[BUILDINGSECURE]
Kenneth P. Birman
Building Secure and Reliable Network Applications 1996
Greenwich: Manning

[ECMASCRIPT]
Standard ECMA-262: ECMAScript Language Specification
http://www.ecma-international.org/publications/standards/Ecma-262.htm
Viitattu 2.6.2004

[EMPAINEISTO]
Pentti Routio
Empiirisen aineiston kerääminen
http://www2.uiah.fi/projects/metodi/060.htm
Viitattu 2.11.2003

[FORMUPLOAD]
Network Working Group
E. Nebel et al.
RFC 1867 - Form-based File Upload in HTML
http://www.faqs.org/rfcs/rfc1867.html
Viitattu 1.6.2004

[GRAPHML]
U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M.S. Marshall 2002
GraphML Progress Report: Structural Layer Proposal.
Springer-Verlag

[HTML4.1]
Dave Raggett, Arnaud Le Hors, Ian Jacobs
W3C
HTML 4.01 Specification
http://www.w3.org/TR/html4/
Viitattu 1.5.2004

[HTTP/1.1]
Network Working Group
R. Fielding et al.
Hypertext Transfer Protocol -- HTTP/1.1 RFC 2616
http://www.w3.org/Protocols/rfc2616/rfc2616.html
Viitattu 1.6.2004

[HCI]
Kari-Jouko Räihä, Saila Ovaska
Ihmisen ja tietokoneen vuorovaikutus / Käyttöliittymien arviointimenetelmiä
http://www.cs.uta.fi/~ov/itv/luennot/kalvot/arviointi/
Viitattu 2.11.2003

[INTERNETWORKING]
Douglas E. Comer 2000
Internetworking with TCP/IP Principles, Protocols, and Architectures
New Jersey: Prentice-Hall, Inc.

[IPSUITE]
Internet Protocol Suite
http://en.wikipedia.org/wiki/Internet_protocol_suite
Wikipedia
Viitattu: 15.9.2004

[KOGNLÄPI]
Marko Nieminen, Teknillinen korkeakoulu, työpsykologian ja johtamisen laboratorio käytettävyystutkimusryhmä
Kognitiivinen läpikäynti
http://www.vtt.fi/aut/rm/val45/usabil2/loppurap/lrap21.htm
Viitattu 2.11.2003

[KÄYTARVIO]
Antti Kokkonen, Aino Ahtinen, 2000
Käytettävyyden merkitys verkko-oppimisessa
http://www.cs.uta.fi/~ak60389/IPOPP2000/kaytettavyys_2.html
Tampereen yliopisto, tietojenkäsittelytieteiden laitos
Viitattu 2.11.2003

[LOKIANALYSOINTI]
Log analyzers Comparisons
http://awstats.sourceforge.net/docs/awstats_compare.html
AWStats
Viitattu 14.10.2004

[LOKIMUODOT]
LOGFILE FORMATS
http://www.http-analyze.org/manual2.4/man01.html
Impressum RENT-A-GURU
Viitattu 14.10.2004

[MEANTIME]
Martin Pool 2000
Meantime: non-consensual http user tracking using caches
http://sourcefrog.net/projects/meantime/
Viitattu 10.1.2004

[MIME]
RFC 2045 - Multipurpose Internet Mail Extensions (MIME)
Network Working Group
N. Freed et al.
http://www.faqs.org/rfcs/rfc2045.html
Viitattu 30.10.2004

[NCSALOKI]
Logging Control In W3C httpd - The Common Logfile Format
http://www.w3.org/Daemon/User/Config/Logging.html
W3C
Viitattu 13.10.2004

[REDSHERIFF]
Comprehensive Web Analytics
http://www.redsheriff.com
NetRatings, Inc
Viitattu 14.10.2004

[SOCKET]
Unix Socket FAQ
http://www.developerweb.net/forum/
Viitattu 13.10.2004

[USABILITYENGINEERING]
Xristine Faulkner 2000
Usability Engineering – Grassroots Series
New York: Palgrave

[USABILITYEVA]
Sirpa Riihiaho 2000
Experiences with Usability Evaluation Methods
Helsinki University of Technology, Laboratory of Information Processing Science

[USABILITYMET]
Kristiina Karvonen, 2003
Usability methods
http://www.cs.hut.fi/Opinnot/T-106.850/PMRG/s2003/materiaali/luennot/Experimentation_usability.pdf
Helsinki University of Technology, Product Modelling & Realisation Group
Viitattu 2.11.2003

[WBI]
Barrett, R. & Maglio, P. P.
Web Intermediaries (WBI)
http://www.almaden.ibm.com/cs/wbi/
IBM
Viitattu 11.10.2004

[WEBKÄYT]
Kai Öörni 2000
Web-sivut ja niiden käytettävyys
Oulun yliopisto – Informaatiotutkimus

[XHTML]
HTML Working Group
Steven Pemberton et al.
XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)
http://www.w3.org/TR/xhtml1/
Viitattu 2.10.2004

[YED]
yWorks
yEd – Java™ Graph Editor
yEd is a powerful graph editor that is written entirely in the JavaTM programming language.
http://www.yworks.com/en/products_yed_about.htm
Viitattu 7.10.2004

LIITE 1: HTTP-PROTOKOLLAN TILAKOODIT

HTTP-protokollan versiossa 1.1 [HTTP/1.1 kappale 10.1] on palvelupyynnön vastauksien tilakoodit on jaoteltu viiteen eri pääluokkaan. Tilakoodit ovat kolminumeroisia ja niitä seuraa selkokielinen versio tilakoodista.

Alla olevissa taulukoissa on esitelty tilakoodiluokat sekä jokainen tilakoodi selityksineen.

Luokka	Mahdolliset tilakoodit
Ilmoitusluontoiset	100-199
Palvelupyyntö onnistunut	200-299
Uudelleenohjaus	300-399
Asiakasohjelmiston virhe	400-499
Palvelinohjelmiston virhe	500-599

Taulukko 20: HTTP:n palvelupyynnön vastauksien tilakoodien luokat.

Tilakoodi	Selvennys
100 Continue	Selaimen tulisi jatkaa palvelupyyntöään. Palvelin on vastaanottanut alkuperäisen palvelupyynnön ja hyväksynyt sen.
101 Switching Protocols	Palvelin on ymmärtänyt ja hyväksynyt selaimen pyynnön vaihtaa sovelluksen tiedonsiirtoon käyttämää protokollaa.

Taulukko 21: Ilmoitusluontoiset tilakoodit.

Tilakoodi	Selvennys
200 OK	Palvelupyyntö oli onnistunut. Kun palvelin palauttaa normaalisti haetun resurssin palautetaan selaimelle tämä tilakoodi.
201 Created	Palvelupyyntö on toteutettu ja uusi resurssi on luotu.
202 Accepted	Palvelupyyntö on hyväksytty suoritettavaksi, mutta se ei ole vielä valmis.
203 Non-Authoritative Information	Palautettu vastaus ei ole peräisin palvelimelta jolle palvelupyyntö osoitettiin. Vastaus voi olla peräisi välimuistista tai kolmannen osapuolen kopiosta.
204 No Content	Palvelupyyntö on suoritettu, mutta vastaus ei sisällä mitään.
205 Reset Content	Palvelupyyntö on suoritettu, mutta selaimen tulisi tyhjentää näkymä (HTML-lomake) josta kyseenomainen palvelupyyntö on muodostettu.
206 Partial Content	Palvelupyyntö on suoritettu ja vastaus palautetaan selaimelle erillisinä HTTP-viesteinä. Palvelupyynnössä on ilmoitettava `Range`-otsikko.

Taulukko 22: Palvelupyynnön onnistuessa palautettavat tilakoodit.

Tilakoodi	Selvennys
300 Multiple Choices	Haettu resurssi vastaa useampaa resurssia palvelimella.
301 Moved Permanently	Haettu resurssi on siirretty pysyvästi toiseen URL-osoitteeseen. Jatkossa selaimen tulisi hakea resurssi suoraan uudesta osoitteesta.
302 Found	Haettu resurssi löytyi eri URL-osoitteesta, mutta jatkossa tulisi resurssin löytämiseksi käyttää alkuperäistä URL-osoitetta.
303 See Other	Haettu resurssi voi löytyä toisesta osoitteesta, ja selaimen tulisi hakea se GET-palvelupyynnöllä toisesta osoitteesta.
304 Not Modified	Haettu resurssi ei ole muuttunut edellisen vierailun jälkeen. Tämä tyypin HTTP-vastaus ei saa sisältää sisältöä.
305 Use Proxy	Selaimen on käytettävä välityspalvelinta joka on ilmoitettu vastauksen `Location` HTTP-otsikossa.
306 (Unused)	Käytetty HTTP:n vanhoissa versiossa. Tilakoodi on varattu tästä johtuen.
307 Temporary Redirect	Haettu resurssi sijaitsee tilapäisesti toisessa osoitteessa.

Taulukko 23: Uudelleenohjauksen yhteydessä palautettava tilakoodit.

Tilakoodi	Selvennys
400 Bad Request	Palvelin ei kyennyt ymmärtämään selaimen lähettämään palvelupyyntöä.
401 Unauthorized	Haettu resurssi vaatii kirjautumisen.
402 Payment Required	Tilakoodi varattu tulevaisuuden käyttöä varten.
403 Forbidden	Palvelin ymmärsi palvelupyynnön, mutta kieltäytyy suorittamasta sitä.
404 Not Found	Haettua resurssia ei löytynyt.
405 Method Not Allowed	Palvelupyynnön muoto (GET, POST, HEAD jne) ei ole sallittu.
406 Not Acceptable	Selaimen ilmoittaa palvelupyynnössä sen tukemat tiedostomuodot. Jos haettu resurssi ei ole jotain näistä muodoista palautetaan tämä otsikko.
407 Proxy Authentication Required	Selaimen on kirjauduttava välityspalvelimelle päästäkseen käyttämään haettu resurssia.
408 Request Timeout	Selain ei tuottanut palvelupyyntöä toivotussa ajassa jolloin palvelin katkaisi yhteyden ja palautti tämän tilakoodin.
409 Conflict	Resurssia ei voitu palauttaa johtuen resurssin nykyisestä tilasta. Tilakoodi palautetaan kun voidaan olettaa, että selain voi ongelman korjata.
410 Gone	Resurssia ei ole enää palvelimella ja uutta sijaintia ei tiedetä.
411 Length Required	Palvelin ei suostu suorittamaan palvelupyyntöä ellei `Content-Length` HTTP-otsikkoa ole määrätty palvelupyynnössä.
412 Precondition Failed	Palvelupyynnössä välitetyissä HTTP-otsikoissa määritetyistä ehdoista jokin tai jotkin eivät toteutuneet.
413 Request Entity Too Large	Palvelupyyntö oli liian suuri suoritettavaksi.
414 Request-URI Too Long	URL-osoite oli liian pitkä.
415 Unsupported Media Type	Resurssin mediatyyppiä ei tueta.
416 Requested Range Not Satisfiable	Palvelupyynnössä määritetty `Range` HTTP-otsikko oli virheellinen.
417 Expectation Failed	Odotusarvo joka määritettiin `Expect` HTTP-otsikossa ei toteutunut.

Taulukko 24: Asiakasohjelmiston virheen yhteydessä palautettavat tilakoodit.

Tilakoodi	Selvennys
500 Internal Server Error	Palvelimella tapahtui odottamaton virhe.
501 Not Implemented	Palvelimelle ei ole toteutettu toiminnallisuutta täyttämään palvelupyynnön määrittämiä toimintoja.
502 Bad Gateway	Tiedon välittäjän toimiva palvelin vastaanotti virheellisen vastauksen yrittäessään suorittaa palvelupyyntöä.
503 Service Unavailable	Palvelupyyntöä ei voitu suorittaa, koska palvelin ei tilapäisesti voi suorittaa sitä. Tämä saattaa johtua esimerkiksi palvelun ruuhkautumisesta.
504 Gateway Timeout	Välittäjän toimiva palvelin ei tuottanut palvelupyyntöä toivotussa ajassa jolloin palvelin katkaisee yhteyden ja palauttaa tämän tilakoodin.
505 HTTP Version Not Supported	Palvelin ei tue HTTP:n versiota joka on määritetty palvelupyynnössä.

Taulukko 25: Palvelinohjelmiston virheen yhteydessä palautettavat tilakoodit.